Department of Computer Applications

Details of Lesson Plan

S.No.	Particulars	Details
1.	Course Name	Object Oriented Programming Using Java
2.	Course Code	PSCSATC 223
3.	Academic Year	2023-2024
4.	Semester	2nd
5.	Number of Lesson plans	45
6.	Faculty Assigned	Mr. Vishal Gupta

Faculty Signature

Lesson Plan No. 1	**Course Name:** Object Oriented Programming Using Java **Topic:** : Features, Object Oriented concepts, Java Virtual Machine Concepts	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the core features of Java programming language. • Grasp the fundamental concepts of Object-Oriented Programming (OOP). • Acquire basic knowledge of the Java Virtual Machine (JVM).
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Briefly introduce Java as a platform-independent, high-level programming language. - Highlight the importance of Java in today's software development landscape. - Create interest by mentioning real-world applications of Java (e.g., Android apps, enterprise systems). 2. **Development** (30 minutes) **a. Java Features** - Platform independence (Write Once, Run Anywhere - WORA). - Automatic memory management (garbage collection). - Strong type system. - Rich standard library. b. **Object-Oriented Concepts** - Introduction to classes and objects. - Encapsulation, inheritance, polymorphism, and abstraction. - Explain these concepts with simple examples **c. Java Virtual Machine (JVM)** - Explain the role of the JVM in executing Java byte code. - Briefly touch upon the JVM architecture (classloader, runtime data area, execution engine). **3. Exercise (5 minutes) –** - What is the core concept that allows Java programs to run on different platforms without recompilation? - Name the four fundamental principles of Object-Oriented Programming. - What is the role of the Java Virtual Machine in the Java programming process? Learning Resources for Students (Public Cloud)
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.

	2. Suggested Reading - Features, Object Oriented concepts, Java Virtual Machine Concepts - https://docs.oracle.com/javase/tutorial/ **3. Homework** - Write a short paragraph on one of the Java features or OOP concepts covered in class. - Try to create a simple Java program demonstrating the concept of a class and object Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2	**Course Name:** Object Oriented Programming Using Java **Topic:** : Primitive Data Type and Variables	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of variables and data types. • Differentiate between different primitive data types. • Learn how to declare and initialize variables.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Start with a real-world analogy. Explain how variables are like labeled boxes that can hold different types of data. Introduce the concept of data types as the type of information that can be stored in a variable. 2. **Development** (30 minutes) **a. Primitive Data Types:** - Explain the concept of primitive data types as the basic building blocks of data. - Discuss the common primitive data types: int, float, char, Boolean. - Provide examples of each data type and their usage. **b. Variables** - Define variables as named storage locations for data. - Explain how to declare and initialize variables. - Demonstrate variable usage with simple **c. Data Type Conversion** - Briefly introduce implicit and explicit type conversions. - Provide simple examples of type casting. **3. Exercise (5 minutes) –** - What is a variable in programming? - What is the difference between an integer and a float? - Give an example of a boolean data type..
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading - Primitive Data Type and Variables https://docs.oracle.com/javase/tutorial/ 3. Homework -Practice declaring and initializing variables of different data types. -Write short programs to perform arithmetic operations on different data types.

	-Explore more about data type conversion and its applications. Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3	**Course Name:** Object Oriented Programming Using Java **Topic:** : Java Keywords, Java Operators	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of keywords and their significance in Java. • Differentiate between different types of operators in Java. • Learn the basic syntax and usage of operators..
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by briefly explaining what programming is and its importance in today's world. - Introduce the concept of Java as a popular programming language. - Highlight the role of keywords and operators in Java programming. 2. **Development** (30 minutes) **a. Java Keywords:** - Define keywords as reserved words with predefined meanings in Java. - Explain the importance of keywords in program structure and control flow. - Provide examples of common keywords like class, public, static, void, int, double, etc. **b. Java Operators:** - Define operators as symbols used to perform operations on variables and values. - Categorize operators into arithmetic, relational, logical, assignment, unary, and ternary operators. - Provide simple examples for each category with basic calculations and comparisons. **c.** Operator Precedence**:** - Briefly explain the concept of operator precedence and associativity. - Give examples to illustrate how operators are evaluated in expressions. **3. Exercise (5 minutes) –** - What is the difference between a keyword and an identifier in Java? - Give examples of three arithmetic operators and their functions. - What is the output of the following expression: `5 + 3 * 2`?
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading

	Java Keywords, Java Operators https://docs.oracle.com/javase/tutorial/ 3. Homework -Practice writing simple Java programs using different operators. - Create a list of commonly used keywords and their functions. - Research and understand the concept of operator overloading. Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4	**Course Name:** Object Oriented Programming Using Java **Topic:** : Expressions in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of expressions in Java. • Differentiate between expressions and statements. • Learn about different types of expressions (arithmetic, relational, logical, assignment, and conditional).
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Start by asking students about mathematical expressions they have encountered. - Relate this to computer programming and introduce the concept of expressions in Java. - Explain that expressions are combinations of values, variables, operators, and method calls that produce a result. 2. **Development** (30 minutes) **a. Arithmetic Expressions:** - Explain the arithmetic operators (+, -, *, /, %). - Demonstrate examples of arithmetic expressions and their evaluation. - Discuss operator precedence and associativity. **b. Relational Expressions:** - Introduce relational operators (==, !=, <, >, <=, >=). - Explain how relational expressions evaluate to boolean values. - Provide examples of comparison expressions. **c. Logical Expressions:** • Introduce logical operators (&&, ||, !). • Explain how logical expressions combine boolean values. • Demonstrate examples of logical expressions. **3. Exercise (5 minutes)** – - What is the difference between an expression and a statement in Java? - Evaluate the following expression: 5 + 3 * 2. - What is the output of the following expression: (5 > 3) && !(2 == 4)??
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading Java Expressions https://stackoverflow.com/questions/22369579/oracle-java-tree-

	tutorial-where-and-how-to-implement-the-tutorial-code 3. Homework - Practice writing different types of expressions. - Create a Java program to calculate the area and perimeter of a rectangle using expressions. - Solve simple logical puzzles using expressions.. Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5	**Course Name:** Object Oriented Programming Using Java **Topic:** : Control Statements in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of control flow in programming. • Learn about different types of control statements (if-else, loops). • Apply control statements to solve simple programming problems.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by asking students if they've ever faced situations where they had to make decisions based on certain conditions. - Explain how computers also make decisions using control statements. - Introduce the concept of control flow in programming. 2. **Development** (30 minutes) **a. If-else statements:** - Explain the syntax and working of if-else statements. - Provide examples of how if-else statements can be used to make decisions. - Demonstrate a simple program with an if-else statement. **b. Loops:** - Introduce the concept of repetition using loops. - Explain the difference between while and for loops. - Provide examples of when to use each type of loop. - Demonstrate simple programs using both while and for loops. **3. Exercise (5 minutes) –** - What is the purpose of an if-else statement? - What is the difference between a while loop and a for loop? - Write a simple if-else statement to check if a number is even or odd.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading Statements in Java https://stackoverflow.com/questions/22369579/oracle-java-tree-tutorial-where-and-how-to-implement-the-tutorial-code 3. Homework - Write a program to find the factorial of a number using a loop. - Write a program to check if a number is prime using an if-

	else statement. - Spend 5 minutes to wrap up and consolidate the learnings
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 6	**Course Name:** Object Oriented Programming Using Java **Topic:** : Arrays in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of arrays in Java. • Learn how to declare, initialize, and access array elements. • Explore different ways to manipulate array elements.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by asking students if they have encountered collections of data in real life (e.g., a list of names, a series of numbers). - Explain that arrays are a fundamental data structure in programming for storing and managing collections of similar data types. - Briefly introduce the concept of an array as a container of elements with the same data type. 2. **Development** (30 minutes) **a. Array Declaration and Initialization**: - Explain how to declare an array specifying its data type and size. - Demonstrate different ways to initialize an array: • Using an array initialize • Using a loop b. **Accessing Array Elements:** - Explain how to access individual elements using their index. - Demonstrate how to iterate through an array using a loop. c. **Array Manipulation:** - Introduce basic array operations like finding the length, modifying elements, and sorting arrays (if time permits). **3. Exercise (5 minutes) –** - What is an array in Java? - How do you access the third element of an array named 'numbers'? - What is the output of the following code
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading Arrays in Java https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

	3. Homework - Create a Java program to store and display the names of 5 students using an array. - Write a Java program to find the average of numbers stored in an array.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 7	**Course Name:** Object Oriented Programming Using Java **Topic:** : Classes and Objects in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concepts of classes and objects in Java. • Learn how to create classes and objects. • Differentiate between class variables, instance variables, and local variables..
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by explaining the real-world analogy of a blueprint and a house. A blueprint (class) is a design for a house (object). - In Java, a class is a blueprint for creating objects. Objects are instances of a class. 2. **Development** (30 minutes) **a. Classes:** - Define a class as a template for creating objects. - Explain class components: fields (variables), methods (functions), and constructors. - Provide a simple example of a `Car` class with properties like color, model, and methods like start(), stop(). **b. Objects:** - Explain that objects are instances of a class. - Demonstrate how to create objects using the `new` keyword. - Access object's properties and invoke methods. c. **Class and Instance Variables:** - Differentiate between class variables (shared by all objects) and instance variables (unique for each object). - Provide examples to illustrate the difference. **3. Exercise (5 minutes) –** - What is the difference between a class and an object? - How do you create an object in Java? - Explain the concept of instance variables.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading Classes and Objects in Java https://stackoverflow.com/questions/22369579/oracle-java-tree-tutorial-where-and-how-to-implement-the-tutorial-code

	3. Homework - Create a class named `Student` with attributes like name, roll number, and age. - Create two objects of the `Student` class and assign values to their attributes. - Write a method to display student information.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 8	**Course Name:** Object Oriented Programming Using Java **Topic:** : Static Methods	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of static methods in programming. • Differentiate between static and instance methods. • Learn to create and use static methods effectively.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by asking students if they have encountered any situations where they needed to perform actions without creating an object. - Introduce the concept of static methods as a way to achieve this. - Explain that static methods belong to the class and not to any specific object. 2. **Development** (30 minutes) **a. Defining Static Methods:** - Explain the syntax for declaring a static method using the static keyword. - Demonstrate how static methods can access class variables and other static methods. **b. Accessing Static Methods:** - Show how to call static methods directly using the class name without creating an object. - Discuss the use cases of static methods, such as utility methods, factory methods, and main methods. **c. Advantages and Disadvantages:** - Explain the benefits of using static methods, such as efficiency and code organization. - Discuss the potential drawbacks, such as reduced flexibility and testability. **d. Exercise (5 minutes) –** - What is the keyword used to declare a static method? - Can a static method access instance variables? - When would you use a static method instead of an instance method?
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading Static Methods https://stackify.com/java-tutorials/

	3. Homework - Create a Java class with at least one static method and one instance method. - Explain the difference between the two methods in comments..
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 9	**Course Name:** Object Oriented Programming Using Java **Topic:** : Constructors	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of constructors in Java. • Learn the different types of constructors. • Be able to create and use constructors in Java programs..
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Briefly explain the concept of objects in Java. - Introduce the idea of initializing object data when creating an object. - Explain that constructors are special methods used for this purpose. 2. **Development** (30 minutes) **a. Default constructor:** - Explain that every class has a default constructor if no other constructors are defined. - Demonstrate how to create a simple class with a default constructor. **b. Parameterized constructors:** - Explain the concept of passing parameters to a constructor. - Demonstrate how to create a class with parameterized constructors. - Show how to use parameterized constructors to initialize object data. **c. Constructor overloading:** - Explain the concept of constructor overloading. - Demonstrate how to create multiple constructors with different parameter lists. - Discuss the importance of constructor overloading for flexibility. **3. Exercise (5 minutes) –** - What is a constructor in Java? - Explain the difference between a default constructor and a parameterized constructor. - What is constructor overloading?
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading Constructors https://stackoverflow.com/questions/22369579/oracle-java-tree-

	tutorial-where-and-how-to-implement-the-tutorial-code 3. Homework - Create a Java class with at least two constructors (default and parameterized). - Write a program to create objects using both constructors and print their values.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 10	**Course Name:** Object Oriented Programming Using Java **Topic:** : Method Overloading	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of method overloading. • Differentiate between method overloading and overriding. • Be able to create overloaded methods in Java.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by asking students if they have ever used the same word with different meanings in different contexts. - Explain that method overloading is a similar concept in Java, where multiple methods can have the same name but different parameters. - Provide a real-world analogy, such as a calculator with different buttons for addition based on the number of operands (e.g., add two numbers, add three numbers). 2. **Development** (30 minutes) **a. Definition and Syntax:** - Explain method overloading: multiple methods with the same name but different parameter lists. - Demonstrate the syntax of method overloading with examples. **b. Overloading Rules:** - Discuss the rules for method overloading: number of parameters, data types of parameters, and order of parameters must differ. - Provide examples to illustrate valid and invalid overloading scenarios. **c. Overloading vs. Overriding:** - Differentiate between method overloading (same class, different parameters) and method overriding (subclass, same method signature). - Use a diagram or table to compare the two concepts. **3. Exercise (5 minutes) –** - What is method overloading in Java? - What are the rules for method overloading? - Differentiate between method overloading and overriding.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading Method Overloading

	https://docs.oracle.com/javase/tutorial/ 3. Homework - Create a Java program with at least three overloaded methods. - Explain the concept of method overloading to a classmate.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 11	**Course Name:** Object Oriented Programming Using Java **Topic:** : Inheritance	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of inheritance in object-oriented programming. • Identify the base and derived classes in an inheritance hierarchy. • Explain the benefits and use cases of inheritance.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by asking students if they have any family resemblances with their parents or siblings. Use this analogy to introduce the concept of inheritance in programming. - Explain that inheritance is a fundamental concept in object-oriented programming (OOP) where one class inherits properties and methods from another class. 2. **Development** (30 minutes) **a. Base Class and Derived Class:** - Define inheritance as a mechanism for creating new classes (subclasses) from existing ones (superclasses). - Explain the extends keyword and its usage. - Demonstrate with a simple example of a Animal class and its subclasses like Dog and Cat. **b. Types of Inheritance:** - Briefly introduce single, multilevel, hierarchical, and hybrid inheritance with examples. - Focus on single inheritance for this lesson, explaining it in detail. **c. Inheritance and Methods:** - Explain method overriding and super keyword - Demonstrate how to override methods in subclasses. **3. Exercise (5 minutes) –** - What is the keyword used to create a subclass in Java? - Explain the difference between a superclass and a subclass.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading Inheritance https://docs.oracle.com/javase/tutorial/java/concepts/inheritance.html 3. Homework - Create a Java program demonstrating single inheritance with

	two classes: Shape as the superclass and Rectangle as the subclass. The Rectangle class should inherit properties from Shape and have additional methods to calculate area and perimeter.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 12	**Course Name:** Object Oriented Programming Using Java **Topic:** : Access Control in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of access control in Java. • Learn about different access modifiers (public, private, protected, default). • Apply access control modifiers to classes, methods, and variables.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Briefly explain the importance of data security and encapsulation in object-oriented programming. - Introduce the concept of access control as a mechanism to protect data and methods from unauthorized access. 2. **Development** (30 minutes) **a. Access Modifiers:** - Explain the four access modifiers: public, private, protected, and default. - Discuss the accessibility levels of each modifier within and outside a class, package, and subclass. **b. Applying Access Control:** - Demonstrate how to use access modifiers with classes, methods, and variables. - Provide examples of how to control access to data members and methods. **c. Best Practices:** - Discuss the importance of proper access control for code maintainability and security. - Provide guidelines for choosing the appropriate access modifier. **1. Exercise (5 minutes) –** - What is the most restrictive access modifier in Java? - Can a class be declared as private? Explain. - What is the default access level if no access modifier is specified?
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading Access Control https://docs.oracle.com/en/java/javase/11/security/java-security-overview1.html

	3. Homework - Create a Java class with different access modifiers for data members and methods. - Experiment with different combinations of access modifiers and observe the results. - Write a short explanation of the difference between public and protected access.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 13	**Course Name:** Object Oriented Programming Using Java **Topic:** : Method Overriding in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of method overriding in Java. • Differentiate between method overloading and overriding. • Learn the rules and syntax for method overriding. • Apply method overriding in practical programming examples.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by asking students if they remember method overloading. Briefly recap the concept. - Introduce the idea of inheritance and how it relates to method overriding. - Explain that method overriding allows subclasses to provide their own implementation of a method inherited from a superclass. 2. **Development** (30 minutes) **a. Concept of Method Overriding:** - Explain that method overriding occurs when a subclass provides a specific implementation of a method that is already defined in its parent class. - Demonstrate with a simple example **b. Rules for Method Overriding:** - The method in the subclass must have the same name, return type, and parameter list as the method in the superclass. - The access modifier of the overridden method cannot be more restrictive than the original method. **c. Polymorphism and Method Overriding:** - Explain how method overriding is a key aspect of polymorphism. - Demonstrate runtime polymorphism using an example **3. Exercise (5 minutes) –** - What is the difference between method overloading and method overriding? - What are the necessary conditions for method overriding? - How does method overriding support polymorphism?
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading

	3. Homework - Create a Java program demonstrating method overriding with different classes like Animal, Bird, and Fish. - Explain the concept of method overriding.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 14	**Course Name:** Object Oriented Programming Using Java **Topic:** : Garbage Collection in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of garbage collection in Java. • Learn about the garbage collection process. • Identify objects eligible for garbage collection.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by discussing memory management in programming. - Explain the challenges programmers face in manual memory management. - Introduce the concept of automatic memory management. - Lead into the topic of garbage collection as a solution. 2. **Development** (30 minutes) **a. Garbage Collector:** - Explain what a garbage collector is and its role in Java. **b. Garbage Collection Process:** - Describe the basic steps involved in the garbage collection process: marking, sweeping, and compaction. **c. Objects Eligible for Garbage Collection:** - Discuss the concept of object references and how objects become eligible for garbage collection when they are no longer referenced. **3. Exercise (5 minutes) –** - What is garbage collection in Java? - What are the main phases of the garbage collection process? - When is an object eligible for garbage collection?
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html 3. Homework - Write a Java program to demonstrate object creation and garbage collection using the `finalize()` method. - Different garbage collection algorithms used in Java.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 15	**Course Name:** Object Oriented Programming Using Java **Topic:** : Abstract Class in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of abstract classes and its significance in Java. • Differentiate between abstract classes and interfaces. • Learn how to create abstract classes and abstract methods. • Understand the role of abstract classes in inheritance.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by discussing the concept of inheritance and its benefits in code reusability. - Introduce the limitation of abstract methods in regular classes. - Explain the need for a special type of class to handle abstract methods - abstract classes. 2. **Development** (30 minutes) **a. What are abstract classes?:** - Define abstract classes as classes that cannot be instantiated. - Explain that they serve as blueprints for subclasses. - Discuss the use of the `abstract` keyword to declare abstract classes. **b. Abstract methods:** - Define abstract methods as methods without a body. - Explain that abstract methods must be declared within abstract classes. - Demonstrate how abstract methods force subclasses to implement specific behavior. **c. Inheritance with abstract classes:** - Explain how subclasses inherit abstract classes. - Discuss the requirement for subclasses to override abstract methods. - Provide examples of abstract class hierarchies. **3. Exercise (5 minutes) –** - What is an abstract class and why is it used? - Can you instantiate an abstract class? Why or why not? - What is the relationship between abstract classes and abstract methods?
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading

	3.Homework - Create a simple abstract class with at least one abstract method. - Create a subclass that extends the abstract class and implements the abstract method. - Explain the difference between abstract classes and interfaces in your own words.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 16	**Course Name:** Object Oriented Programming Using Java **Topic:** : Packages in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of packages in Java • Learn how to create and use packages • Identify the benefits of using packages in Java
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by discussing the need for organizing code in larger projects. - Explain how packages help in preventing naming conflicts and improving code reusability. - Introduce the concept of a package as a container for classes and interfaces. 2. **Development** (30 minutes) **a. Package declaration:** - Explain how to declare a package using the package keyword. - Demonstrate how to create a package structure in the file system. **b. Importing packages:** - Explain the use of the import statement to access classes from other packages. - Differentiate between single-type import and on-demand import. **c. Package naming conventions:** - Discuss the recommended naming conventions for packages (reverse domain name style). - Emphasize the importance of using meaningful and unique package names. **3. Exercise (5 minutes) –** - What is the purpose of a package in Java? - How do you declare a package in a Java program? - Explain the difference between single-type import and on-demand import.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/tutorial/java/package/index.html 3.Homework - Create a simple Java project with multiple packages.

	- Create at least two classes in different packages. - Import necessary classes and use them in your code.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 17	**Course Name:** Object Oriented Programming Using Java **Topic:** : Interfaces in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: • Understand the concept of interfaces in Java • Learn how to define and implement interfaces • Identify the benefits of using interfaces in Java
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Begin by discussing the need for abstraction in object-oriented programming. - Explain how interfaces provide a blueprint for classes to follow. - Introduce the concept of an interface as a contract that defines methods without implementation. 2. **Development** (30 minutes) **a. Interface definition:** - Explain how to define an interface using the interface keyword. - Demonstrate how to declare methods with method signatures in an interface. **b. Interface implementation:** - Show how classes implement interfaces using the implements keyword. - Explain the requirement to provide implementations for all interface methods in the implementing class. **c. Benefits of interfaces:** - Discuss the advantages of using interfaces, such as polymorphism, achieving abstraction, and loose coupling. **3. Exercise (5 minutes) –** - What is an interface in Java? - How do you define an interface? - What is the relationship between a class and an interface?
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/tutorial/java/concepts/interface.html 3.Homework - Create an interface named Shape with methods area() and perimeter(). - Create two classes, Circle and Rectangle, implementing the

	Shape interface. - Implement the area() and perimeter() methods for both classes.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 18	**Course Name:** Object Oriented Programming Using Java **Topic:** : Exceptions Handling & Types of Exceptions in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: - Understand the concept of exceptions in Java programming - Learn how to handle exceptions using try-catch-finally blocks - Differentiate between checked and unchecked exceptions - Identify common types of exceptions and their causes.
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Start by asking students about the importance of error handling in real-life scenarios (e.g., dividing by zero, accessing an invalid file). - Explain how exceptions are similar to real-life errors and how Java provides mechanisms to handle them. Introduce the concept of exceptions as unexpected events that disrupt the normal flow of a program. 2. **Development** (30 minutes) **a. Exception Handling:** - Explain the try-catch-finally block syntax. - Demonstrate how to handle exceptions using a simple example (e.g., dividing by zero). - Discuss the importance of the finally block for cleanup operations. **b. Types of Exceptions:** - Differentiate between checked and unchecked exceptions. - Provide examples of common checked exceptions (e.g., IOException, FileNotFoundException). - Explain the concept of runtime exceptions and their subclasses (e.g., ArithmeticException, NullPointerException). **c. Common Exceptions:** - Discuss other common exceptions like arrayIndexOutOfBoundsException, numberFormatException, etc. - Provide examples of how these exceptions can occur.. **1. Exercise (5 minutes) –** - What is the purpose of a try-catch block? - Differentiate between checked and unchecked exceptions. - What is the difference between an exception and an error?
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation

	from students on these. 2. Suggested Reading https://docs.oracle.com/javase/tutorial/essential/exceptions/ 3.Homework - Create simple Java programs that demonstrate different types of exceptions and their handling. Research and identify other types of exceptions that can occur in Java applications.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 19	**Course Name:** Object Oriented Programming Using Java **Topic:** : Multithreading in Java	**Course No.: PSCSATC-223**

Objectives	At the end of the lesson the student shall be able to: - Understand the concept of multithreading and its benefits. - Learn how to create and manage threads in Java. - Explore thread synchronization and its importance. - Understand the concept of thread priority and its implications
Teaching Aids (if any)	a. Power Point Presentation
Teaching Development	1. **Introduction** (5 minutes) - Start with a real-world analogy like multitasking on a computer. - Explain how multiple tasks can run simultaneously to improve performance. - Introduce the concept of threads as lightweight processes within a program. 2. **Development** (30 minutes) **a. Thread Creation and Lifecycle:** - Explain the `Thread` class and its methods. - Demonstrate how to create and start a thread. - Discuss the different states of a thread (new, Runnable, running, blocked, waiting, timed_waiting, terminated). b. **Thread Synchronization:** - Introduce the concept of shared resources and race conditions. - Explain the synchronized keyword and its use. - Demonstrate how to use synchronized methods and blocks. c. **Thread Priority:** - Explain the concept of thread priority and its range (1-10). - Demonstrate how to set thread priority using set Priority method. - Discuss the limitations of thread priority and its dependence on the JVM. d. **Thread Communication:** - Briefly introduce wait, notify, and notifyAll methods. - Explain their use in inter-thread communication. **3. Exercise (5 minutes) –** - What is a thread in Java? - How do you create and start a thread? - What is the purpose of thread synchronization? - How can you influence the order of thread execution?
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation

	from students on these. 2. Suggested Reading https://www.baeldung.com/java-thread-priority 3.Homework - Write a Java program to create multiple threads with different priorities and observe their execution order. - Create a simple program to demonstrate thread synchronization using a shared counter and different thread priorities.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 20	**Course Name:** Object Oriented Programming Using Java **Topic:** : Synchronization in Java	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: - Understand the concept of thread safety and its importance. - Learn about race conditions and how synchronization can prevent them. - Explore different synchronization mechanisms in Java.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	**1. Introduction** (5 minutes) - Introduce the concept of multithreading and its potential issues. - Explain how multiple threads accessing shared data can lead to inconsistencies. - Highlight the need for synchronization to protect shared data. **2. Development** (30 minutes) a. **Race Conditions:** - Explain what race conditions are and provide examples. - Demonstrate a simple program to illustrate a race condition. b. **Synchronized Keyword:** - Introduce the synchronized keyword and its purpose. - Explain how synchronized methods and blocks work. - Demonstrate how to use synchronized to prevent race conditions. c. **Other Synchronization Mechanisms:** - Briefly mention other synchronization mechanisms like volatile, atomic classes, and ReentrantLock. **3. Exercise (5 minutes)** - What is a race condition? - How does the `synchronized` keyword help in preventing race conditions? - What are the different ways to achieve synchronization in Java?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/tutorial/essential/concurrency/ 3.Homework - Write a Java program to demonstrate a race condition and then fix it using synchronization. - Research and understand the differences between synchronized and volatile keywords. priorities.	
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.	

	Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 21	**Course Name:** Object Oriented Programming Using Java **Topic:** : I/O in Java in Java	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the basic concepts of Input and Output (I/O) in Java. • Learn how to use the `java.io` package for file handling. • Explore the differences between byte streams and character streams.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Start with a brief overview of I/O operations and their importance in Java programming. - Explain how I/O operations allow Java programs to interact with the outside world (e.g., reading from and writing to files, reading user input from the console). 2. **Development** (30 minutes) **a. Basic I/O Operations:** - Discuss the System. in, System. out, and System.err streams. - Introduce the concept of reading from the console using Scanner and writing to the console using System.out.print() and System.out.println(). **b. Byte Streams vs. Character Streams:** - Explain byte streams using classes like FileInputStream and FileOutputStream for reading and writing binary data. - Explain character streams using classes like FileReader and FileWriter for reading and writing text data. **c. File Handling:** - Demonstrate how to create, read, and write files using the `File` class and associated I/O classes. - Introduce exception handling related to I/O operations, focusing on FileNotFoundException and IOException. 3. **Exercise (5 minutes)** - What is the difference between FileInputStream and FileReader in Java? - How do you handle exceptions when performing file I/O operations in Java? - Write a simple Java code snippet to read a line of text from the console using Scanner.	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/tutorial/essential/io/index.html	

	3.Homework - Write a Java program to read data from a file and display it on the console. - Write a Java program to copy content from one file to another using byte streams.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 22	**Course Name:** Object Oriented Programming Using Java **Topic:** Reading and Writing to Console	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand how to read user input from the console. • Learn how to display output to the console. • Apply these techniques in simple Java programs.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	**1. Introduction** (5 minutes) - Begin with a brief overview of console input and output. - Explain how Java handles user interaction via the console and the importance of these operations in programming. - Highlight real-world scenarios where reading from and writing to the console is essential. **2. Development** (30 minutes) **a. Reading from the Console:** - Introduce Scanner class and demonstrate how to use it to capture user input. Example: Scanner scanner = new Scanner(System.in) **b. Writing to the Console:** - Explain the use of System.out.println() and System.out.print() for displaying messages. Discuss formatting options and best practices. **c. Practical Exercise:** - Guide students through creating a simple Java program that reads a user's name and age from the console and then prints a greeting message. **3. Exercise (5 minutes)** - How do you create a Scanner object to read input from the console? - What is the difference between System.out.println() and System.out.print()? - Write a short code snippet that reads a user's age and prints it out.	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html 3.Homework - Create a Java program that prompts the user for their favorite color and number, then prints a personalized message including both pieces of information.	
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents	

Lesson Plan No. 23	**Course Name:** Object Oriented Programming Using Java **Topic:** Reading and Writing Files	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of file handling in Java. • Learn how to create, write to, and read from files. • Become familiar with the File, FileWriter, FileReader, and BufferedReader classes.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Begin by asking students if they have ever saved data to a file on their computer. - Explain the importance of file handling in applications and how it allows programs to store and retrieve data persistently. - Introduce the concept of file I/O operations in Java. 2. **Development** (30 minutes) **a. File Class:** - Explain the purpose of the File class, demonstrating how to create File objects to represent files and directories. - Discuss methods like createNewFile(), exists(), isDirectory(), etc. **b. Writing to a File:** - Introduce FileWriter and BufferedWriter classes. - Demonstrate how to create a FileWriter object to open a file for writing, and then use BufferedWriter to write text to the file. - Explain the importance of closing the file after writing. c. **Reading from a File:** - Introduce FileReader and BufferedReader classes. - Demonstrate how to create a FileReader object to open a file for reading, and then use BufferedReader to read text from the file line by line. - Explain how to handle potential exceptions during file operations. 3. **Exercise (5 minutes)** • What is the purpose of the File class in Java? • Name two classes used for writing to a file in Java. • How do you read a file line by line in Java?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/tutorial/essential/io/ 3.Homework • Write a Java program to create a text file and write some content to it.	

	- Write another Java program to read the content of the file created in the previous step and display it on the console
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 24	**Course Name:** Object Oriented Programming Using Java **Topic:** Transient and Volatile Modifiers	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of transient and volatile modifiers. • Differentiate between transient and volatile keywords. • Learn the use cases of transient and volatile modifiers in Java programming.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	**1. Introduction** (5 minutes) - Start by briefly explaining the concept of memory management in Java. - Introduce the idea of object serialization and multi-threading as a foundation for understanding the need for transient and volatile modifiers. **2. Development** (30 minutes) **a. Transient Modifier:** - Explain the purpose of the transient modifier. - Discuss how it affects object serialization. - Provide examples of when to use the transient modifier (e.g., sensitive data, large objects). b. **Volatile Modifier:** - Explain the concept of shared memory and caching in multi-threaded environments. - Describe the visibility and ordering guarantees provided by the volatile modifier. - Illustrate the use of volatile for shared variables accessed by multiple threads. c. **Comparison of Transient and Volatile:** - Summarize the key differences between transient and volatile. - Create a table comparing the two modifiers based on purpose, usage, and impact. **3. Exercise (5 minutes)** - What happens to a variable declared as transient during object serialization? - When is the volatile modifier typically used in Java programming? - What is the main difference between transient and volatile modifiers?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://www.tutorialspoint.com/difference-between-volatile-and-transient-in-java	

	3.Homework - Create a Java program demonstrating the use of transient and volatile modifiers. - Explain the concept of "happens-before" relationship in the context of volatile variables. - Find real-world examples of where transient and volatile modifiers are used in Java applications?
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 25	**Course Name:** Object Oriented Programming Using Java **Topic:** The String and String Buffer Class.	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of strings and string manipulation in Java. • Learn the difference between String and StringBuffer classes. • Understand the methods used for string manipulation..	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Begin by asking students if they have encountered strings in their daily lives (e.g., names, addresses, messages). - Explain that strings are fundamental data types in programming and are used to represent textual data. - Introduce the concept of immutability and mutability in the context of strings. 2. **Development** (30 minutes) **a. String Class:** - Explain that String is a class in Java representing immutable sequences of characters. - Discuss the creation of String objects using literals and the `new` keyword. - Demonstrate common string operations like concatenation, length, substring, and comparison. **b. StringBuffer Class:** - Introduce StringBuffer as a mutable sequence of characters. - Explain the advantages of StringBuffer over String for string manipulation. - Demonstrate methods like `append`, `insert`, `delete`, reverse, and capacity. 1. **Exercise (5 minutes)** - What is the difference between a String and a StringBuffer? - How can you concatenate two strings in Java? - Give an example of when you would use a StringBuffer instead of a String.	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html 3.Homework - Practice creating and manipulating strings using both String and StringBuffer. - Write a Java program to reverse a string using StringBuffer.	

	- Explore additional methods of the StringBuffer class.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 26	**Course Name:** Object Oriented Programming Using Java **Topic:** The Applet Class	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of applets and their role in Java. • Learn about the basic structure of an applet. • Identify the important methods of the Applet class.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Start by explaining what applets are: small Java programs that run within a web browser. - Discuss their historical significance and why they are less commonly used today. - Briefly touch upon the concept of a web browser as a container for applets. 2. **Development** (30 minutes) a. **Applet Lifecycle:** - Explain the different stages of an applet's life: init(), start(), stop(), destroy(). - Provide simple examples of what happens in each stage. b. **Applet Methods:** - Discuss commonly used Applet methods like paint(), repaint(), resize(), and getParameter(). - Explain their purpose and usage with simple examples. c. **TML Applet Tag:** - Demonstrate how to embed an applet in an HTML page using the <applet> tag. - Explain the attributes of the <applet> tag, such as code, width, height, and archive. 3. **Exercise (5 minutes)** - What are the four main stages in an applet's lifecycle? - What is the purpose of the paint() method in an applet? - How do you embed an applet in an HTML page?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://www.oracle.com/topics/technologies/newtojava/programming-center.html 3.Homework - Create a simple applet that displays a "Hello, World!" message. - Experiment with different applet methods to understand their behavior. - Give the reasons for the decline in applet usage and the alternatives available today.	

Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 27	**Course Name:** Object Oriented Programming Using Java **Topic:** An Applet Skeleton	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the basic structure of an applet. • Identify the essential components of an applet skeleton. • Create a simple applet framework.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	**1. Introduction** (5 minutes) - Begin by explaining what applets are: small Java programs that can be embedded in web pages and run within a web browser. - Highlight the importance of understanding the applet structure for creating interactive web content. **2. Development** (30 minutes) a. **Applet Skeleton Structure:** - Explain the basic structure of an applet class, including the necessary import statements (import java.applet.*; import java.awt.*;) and the extension of the Applet class. - Discuss the purpose of the init(), start(), paint(), and stop() methods. - Provide a simple code example of an empty applet skeleton. b. **Essential Components:** - Explain the role of the init() method for initializing the applet. - Discuss how the start() method is used to start the applet's execution. - Emphasize the importance of the paint() method for drawing graphics on the applet's canvas. - Explain the stop() method for stopping the applet's execution. c. **Creating a Simple Applet Framework:** - Guide students through creating a basic applet project in their IDE. - Demonstrate how to add the necessary code for an applet skeleton. - Encourage students to experiment with different applet components. **3. Exercise (5 minutes)** - What are the four main methods in an applet lifecycle? - What is the purpose of the `paint()` method? - What is the base class for all applets?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.	

	2. Suggested Reading https://www.oracle.com/topics/technologies/newtojava/programming-center.html 3.Homework -Create a simple applet that displays a "Hello, World!" message. -Experiment with different colors and fonts in the `paint()` method.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 28	**Course Name:** Object Oriented Programming Using Java **Topic:** Adding images & sound in Applets	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of multimedia integration in Java. • Learn how to load and display images using Java's AWT and Swing libraries. • Explore playing audio files using Java's built-in classes.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	**1. Introduction** (5 minutes) - Begin by asking students if they've ever seen games or applications with images and sound. - Explain how these elements enhance user experience. - Briefly introduce the concept of multimedia programming in Java. **2. Development** (30 minutes) **a. Image Handling:** - Introduce the ImageIcon class and its usage for loading images. - Demonstrate how to create a JLabel to display an image. - Explain how to resize and position images using image manipulation techniques. **b. Audio Handling:** - Introduce the Clip class from the javax.sound.sampled package. - Demonstrate how to load and play audio files using the AudioInputStream and AudioSystem classes. - Discuss basic audio controls like play, pause, and stop. **3. Exercise (5 minutes)** - What is the primary class used for loading images in Java? - How do you create a visual component to display an image? - Which class is essential for playing audio clips in Java?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading 3.Homework - Create a simple Java application that displays a static image and plays a background music. - Experiment with different image formats (e.g., PNG, JPEG) and audio formats (e.g., MP3, WAV).	
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents	

Lesson Plan No. 29	**Course Name:** Object Oriented Programming Using Java **Topic:** Passing parameters to an applet	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of passing parameters to an applet. • Learn how to use the `<param>` tag to pass parameters. • Utilize the `getParameter()` method to retrieve parameters in the applet.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Begin by briefly revisiting applets and their purpose. - Explain the need for dynamic behavior in applets. - Introduce the concept of passing parameters to customize applet behavior. 2. **Development** (30 minutes) **a. The <param> tag:** - Explain the syntax and attributes of the <param> tag. - Demonstrate how to embed the <param> tag within the <applet> tag. - Provide examples of different parameter types (string, int, etc.). **b. The getParameter() method:** - Explain the purpose of the getParameter() method. - Demonstrate how to use the getParameter() method to retrieve parameter values. - Discuss potential exceptions and error handling. **c. Practical Example:** - Create a simple applet that displays a message based on a passed parameter. - Step-by-step code demonstration. - Explain the code logic and output. 3. **Exercise (5 minutes)** - What is the purpose of the `<param>` tag in an applet? - How do you retrieve a parameter value within an applet? - What happens if a parameter is not found using `getParameter()`?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/tutorial/ 3.Homework - Create an applet that displays a welcome message customized with the user's name as a parameter. - Experiment with different parameter types and their usage. - Research advanced parameter handling techniques (e.g., default values, validation).	

Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 30	**Course Name:** Object Oriented Programming Using Java **Topic:** AWT Components	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of AWT components. • Learn about different types of AWT components. • Create simple GUI applications using AWT components.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - **Brief** overview of Graphical User Interface (GUI). - Importance of GUI applications in software development. - Introduction to the Abstract Window Toolkit (AWT) as a Java library for creating GUI components. 2. **Development** (30 minutes) a. **Basic AWT Components:** - Explain the concept of containers and components. - Discuss the hierarchy of AWT components. - Introduce core containers: Frame, Panel, and Dialog. b. **Common AWT Components:** - Describe basic components like Label, Button, TextField, and TextArea. - Demonstrate how to create and add components to a container. - Explain the concept of event handling (briefly). c. **Layout Managers:** - Introduce the concept of layout managers. - Explain the use of FlowLayout, BorderLayout, and GridLayout. - Demonstrate how to use layout managers to arrange components. 1. **Exercise (5 minutes)** - What is the difference between a container and a component in AWT? - Name three basic AWT components. - What is the purpose of a layout manager?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/awt/package-frame.html 3.Homework - Create a simple GUI application using AWT components. - Experiment with different layout managers. - Explore more AWT components like Checkbox, Choice, and List. -	

Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 31	**Course Name:** Object Oriented Programming Using Java **Topic:** Building User Interface with AWT	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the basic components of AWT for building GUIs. • Learn how to create simple GUI applications using AWT. • Understand the concept of layout managers..	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Brief overview of GUI applications and their importance. - Introduce AWT as a Java toolkit for creating GUIs. - Explain the difference between AWT and Swing (briefly). 2. **Development** (30 minutes) a. **Core AWT Components:** - Explain the concept of containers and components. - Introduce Frame, Panel, and Dialog as core containers. - Demonstrate how to create and display a basic frame. b. **Basic Components:** - Discuss common components like Label, Button, TextField, and TextArea. - Show examples of how to add these components to a frame. c. **Layout Managers:** - Introduce the concept of layout managers. - Explain FlowLayout, BorderLayout, and GridLayout. - Demonstrate how to use layout managers to arrange components within a container. 3. **Exercise (5 minutes)** - What is the difference between a Frame and a Panel in AWT? - Name three basic AWT components. - What is the purpose of a layout manager?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/awt/package-frame.html 3. Homework - Create a simple GUI application using AWT components (e.g., a calculator or a simple form). - Experiment with different layout managers to achieve different layouts. - Explore additional AWT components like Checkbox, Choice, and List.	
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to	

	answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 32	**Course Name:** Object Oriented Programming Using Java **Topic:** Handling Events of Mouse	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of mouse events in Java. • Learn how to create mouse listeners. • Implement mouse event handling in a Java application.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Brief overview of event-driven programming. - Importance of mouse interaction in GUI applications. - Introduction to the MouseListener and MouseMotionListener interfaces. 2. **Development** (30 minutes) a. **MouseListener Interface:** - Explain the methods in MouseListener (click, press, release, enter, exit). - Demonstrate how to implement MouseListener in a class. - Create a simple example to handle mouse clicks. b. **MouseMotionListener Interface:** - Explain the methods in MouseMotionListener (move, drag). - Demonstrate how to implement MouseMotionListener in a class. - Create a simple example to track mouse movement and dragging. c. **Event Handling:** - Explain the concept of event sources and listeners. - Show how to register a listener with a component. - Discuss event propagation and handling. 3. **Exercise (5 minutes)** - What are the two main interfaces for handling mouse events in Java? - What method is called when the mouse button is pressed? - What is the difference between mouseMoved and mouseDragged?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/tutorial/uiswing/events/mouselistener.html 3. Homework - Create a Java application that displays a message when the mouse is clicked on a button. - Create a drawing application that allows users to draw lines by dragging the mouse. - Experiment with other mouse event methods and their combinations.	

Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 33	**Course Name:** Object Oriented Programming Using Java **Topic:** Handling Events of Keyboard	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of event handling in Java. • Learn about keyboard events and their types. • Implement keyboard event listeners to respond to key presses..	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	**1. Introduction** (5 minutes) - Brief overview of event-driven programming. - Importance of event handling in interactive applications. - Introduction to the concept of keyboard events. **2. Development** (30 minutes) **a. Keyboard Events:** - Explain different types of keyboard events (keyPressed, keyReleased, keyTyped). - Discuss the KeyEvent class and its methods. - Demonstrate how to register a key listener to a component. **b. Event Handling:** - Explain the event handling mechanism in Java. - Introduce the ActionListener and KeyListener interfaces. - Demonstrate how to implement event listener methods. **c. Practical Example:** - Create a simple Java application to demonstrate keyboard event handling. - Show how to capture key presses and display corresponding messages. **3. Exercise (5 minutes)** - What are the three main types of keyboard events? - What interface is used to handle keyboard events? - What method is called when a key is pressed?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/tutorial/uiswing/events/keylistener.html 3. Homework - Create a Java program that counts the number of times a specific key is pressed. - Experiment with different keyboard events and their corresponding methods. - Explore advanced keyboard event handling techniques (e.g., key modifiers).	
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.	

Lesson Plan No. 34	**Course Name:** Object Oriented Programming Using Java **Topic:** Event Delegation Model	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the core concepts of the event delegation model. • Identify the components involved in event handling. • Implement simple event handling using listeners and event objects.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Brief overview of event-driven programming. - Contrast with traditional procedural programming. - Introduce the event delegation model as a fundamental concept in GUI programming. 2. **Development** (30 minutes) **a. Components of Event Handling:** - Explain the roles of the event source, event object, and event listener. - Discuss the concept of event registration. - Demonstrate how to create and register event listeners. **b. Event Classes and Methods:** - Introduce common event classes like ActionEvent, MouseEvent, and KeyEvent. - Explain the methods available in event objects for retrieving event information. **c. Event Listener Interfaces:** - Describe the ActionListener interface as an example. - Explain the actionPerformed method and its parameters. - Provide a simple code example to illustrate event handling. 3. **Exercise (5 minutes)** - What are the three main components involved in event handling? - What is the purpose of an event listener? - Describe the general structure of an event handler method.	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://www.geeksforgeeks.org/event-handling-in-java/ 3. Homework - Create a simple Java program that demonstrates event handling for a button click. -Experiment with different event types (e.g., mouse events, keyboard events). - Understand other event listener interfaces beyond ActionListener.	
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to	

	answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

<table>
<tr><td><b>Lesson Plan No. 35</b></td><td><b>Course Name:</b> Object Oriented Programming Using Java
<b>Topic:</b> Layouts and Layout Manager</td><td><b>Course No.: PSCSATC-223</b></td></tr>
<tr><td><b>Objectives</b></td><td colspan="2">At the end of the lesson the student shall be able to:
• Understand the concept of layout managers in Java.
• Learn about different types of layout managers.
• Use layout managers to design effective GUI layouts.</td></tr>
<tr><td><b>Teaching Aids (if any)</b></td><td colspan="2">a. Power Point Presentation</td></tr>
<tr><td><b>Teaching Development</b></td><td colspan="2">1. <b>Introduction</b> (5 minutes)
- Brief overview of GUI design and the importance of layout.
- Explain the problem without layout managers (components overlapping).
- Introduce the concept of layout managers as a solution.
2. <b>Development</b> (30 minutes)
<b>a. Types of Layout Managers:</b>
- Explain FlowLayout, BorderLayout, and GridLayout.
- Discuss the characteristics and usage of each layout manager.
- Provide examples of when to use each layout.
<b>b. Using Layout Managers:</b>
- Demonstrate how to set a layout manager for a container.
- Show how to add components to a container using different layout managers.
- Explain the concept of nested layouts.
<b>c. Common Layout Issues and Solutions:</b>
- Discuss common layout problems (e.g., component resizing, alignment).
- Provide tips for effective layout design.
- Introduce additional layout managers (optional, time permitting).
3. <b>Exercise (5 minutes)</b>
- What is the default layout manager for a JPanel?
- Explain the difference between FlowLayout and GridLayout.
- How do you set a BorderLayout for a JFrame?</td></tr>
<tr><td><b>Closure</b></td><td colspan="2">1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.
2. Suggested Reading
https://docs.oracle.com/javase/tutorial/uiswing/layout/index.html
3. Homework
-Create a simple GUI application using different layout managers.
-Experiment with nested layouts.
-Try to recreate a specific layout design using AWT components and layout managers.</td></tr>
<tr><td><b>Evaluation</b></td><td colspan="2">1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.
Spend 5 minutes to evaluate student assimilation of the lesson contents</td></tr>
</table>

Lesson Plan No. 36	**Course Name:** Object Oriented Programming Using Java **Topic:** JDBC Overview	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of JDBC and its role in database connectivity. • Learn about the basic components of JDBC architecture. • Understand the steps involved in JDBC programming.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Begin by discussing the importance of databases in modern applications. - Introduce the concept of database connectivity and the challenges involved. - Briefly explain how JDBC simplifies database interaction. 2. **Development** (30 minutes) **a. JDBC Architecture:** - Explain the layered architecture of JDBC (JDBC Driver Manager, Driver, Connection, Statement, ResultSet). - Discuss the different types of JDBC drivers (Type 1, 2, 3, 4). **b. JDBC Process:** - Outline the steps involved in JDBC programming (loading the driver, establishing a connection, creating a statement, executing queries, processing results). - Provide a simple code example to illustrate the process. **c. JDBC Statements:** -Introduce the different types of statements (Statement, PreparedStatement, CallableStatement). -Briefly explain the advantages of using prepared statements. 3. **Exercise (5 minutes)** - What is JDBC and what is its primary purpose? - Name the four main components of JDBC architecture. - What are the three types of JDBC statements?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html 3. Homework - Write a Java program to connect to a database (e.g., MySQL, Oracle), create a table, insert some data, and retrieve the data. - Experiment with different types of JDBC statements.	

Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 37	**Course Name:** Object Oriented Programming Using Java **Topic:** JDBC implementation	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the practical aspects of using JDBC to interact with databases. • Learn how to establish a database connection, execute SQL queries, and process results. • Gain experience in handling exceptions and closing database resources.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	**1. Introduction** (5 minutes) - Recap the basic concepts of JDBC architecture and components. - Emphasize the importance of practical implementation for understanding JDBC. - Introduce a simple database scenario (e.g., student database with name, roll number, and marks) **2. Development** (30 minutes) a. **Establishing a Database Connection:** - Demonstrate how to load the appropriate JDBC driver. - Explain the process of creating a database connection using DriverManager.getConnection(). - Discuss connection properties and URL formats. b. **Executing SQL Queries:** - Show how to create Statement and PreparedStatement objects. - Demonstrate executing SELECT, INSERT, UPDATE, and DELETE queries. - Explain handling result sets using ResultSet interface. c. **Error Handling and Resource Management:** - Discuss common JDBC exceptions (e.g., SQLException). - Demonstrate using try-catch blocks to handle exceptions. - Emphasize the importance of closing database resources (Connection, Statement, ResultSet). **3. Exercise (5 minutes)** - What is the first step in JDBC implementation? - How do you create a database connection in JDBC? - What is the purpose of the `ResultSet` interface?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html 3. Homework	

	- Create a Java program to perform CRUD operations (Create, Read, Update, Delete) on a database table of your choice. - Implement error handling and resource closing in your code. - Experiment with different SQL queries and result set processing techniques.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 38	**Course Name:** Object Oriented Programming Using Java **Topic:** Statements, Types of statement objects	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of JDBC statements. • Differentiate between the three types of statement objects: Statement, PreparedStatement, and CallableStatement. • Learn how to use these statement objects to execute SQL queries.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Recap the concept of JDBC and its role in database connectivity. - Introduce the idea that JDBC statements are used to execute SQL queries. - Highlight the importance of choosing the right type of statement object for different scenarios. 2. **Development** (30 minutes) a. **Statement Object:** - Explain the basic functionality of the Statement interface. - Demonstrate how to create a Statement object and execute simple SQL queries. - Discuss the limitations of Statement objects, especially in terms of performance and security. b. **PreparedStatement Object:** - Introduce the concept of precompiled SQL statements. - Explain the benefits of using PreparedStatement objects for performance and security. - Demonstrate how to create a PreparedStatement object, set parameters, and execute queries. c. **CallableStatement Object:** - Introduce the concept of calling stored procedures. - Explain the syntax for calling stored procedures using CallableStatement. - Demonstrate how to register output parameters and retrieve results from stored procedures. 3. **Exercise (5 minutes)** - What are the three types of JDBC statement objects? - When would you use a PreparedStatement instead of a Statement? - How do you register an output parameter in a CallableStatement?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/sql/package-	

	summary.htm 3. Homework - Write Java code to demonstrate the use of each type of statement object (Statement, PreparedStatement, and CallableStatement) with a simple database query. - Experiment with different SQL statements and parameter types for PreparedStatements. - Create a simple stored procedure and call it using a CallableStatement.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 39	**Course Name:** Object Oriented Programming Using Java **Topic:** Statements, Types of statement objects	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of JDBC statements. • Differentiate between the three types of statement objects: Statement, PreparedStatement, and CallableStatement. • Learn how to use these statement objects to execute SQL queries.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Recap the concept of JDBC and its role in database connectivity. - Introduce the idea that JDBC statements are used to execute SQL queries. - Highlight the importance of choosing the right type of statement object for different scenarios. 2. **Development** (30 minutes) a. **Statement Object:** - Explain the basic functionality of the Statement interface. - Demonstrate how to create a Statement object and execute simple SQL queries. - Discuss the limitations of Statement objects, especially in terms of performance and security. b. **PreparedStatement Object:** - Introduce the concept of precompiled SQL statements. - Explain the benefits of using PreparedStatement objects for performance and security. - Demonstrate how to create a PreparedStatement object, set parameters, and execute queries. c. **CallableStatement Object:** - Introduce the concept of calling stored procedures. - Explain the syntax for calling stored procedures using CallableStatement. - Demonstrate how to register output parameters and retrieve results from stored procedures. 3. **Exercise (5 minutes)** - What are the three types of JDBC statement objects? - When would you use a PreparedStatement instead of a Statement? - How do you register an output parameter in a CallableStatement?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/sql/package-	

	summary.htm 3. Homework - Write Java code to demonstrate the use of each type of statement object (Statement, PreparedStatement, and CallableStatement) with a simple database query. - Experiment with different SQL statements and parameter types for PreparedStatements. - Create a simple stored procedure and call it using a CallableStatement.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 40	**Course Name:** Object Oriented Programming Using Java **Topic:** Connection class	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of a JDBC Connection object. • Learn how to establish a database connection using the Connection interface. • Understand the importance of connection properties and methods.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	**1. Introduction** (5 minutes) - Recap the previous lesson on JDBC overview. - Emphasize the importance of establishing a connection to a database for database operations. - Introduce the JDBC Connection class as the fundamental object for this connection. **2. Development** (30 minutes) a. **Connection Interface:** - Explain the Connection interface as the core component for interacting with a database. - Discuss the methods for creating, managing, and closing a database connection. b. **Connection Properties:** - Explain the concept of connection properties (URL, username, password). - Demonstrate how to set connection properties using the DriverManager.getConnection() method. c. **Connection Lifecycle:** - Discuss the importance of proper connection management. - Explain the concept of connection pooling and its benefits. **3. Exercise (5 minutes)** - What is the primary purpose of a JDBC Connection object? - Name the three essential connection properties. - Why is it important to close a database connection after use?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/sql/Connection.html 3. Homework - Write a Java program to establish a connection to a database using the Connection class. - Experiment with different connection properties and error handling.	

Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 41	**Course Name:** Object Oriented Programming Using Java **Topic:** Types of result set	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the concept of result sets in JDBC. • Differentiate between the three main types of result sets: ResultSet.TYPE_FORWARD_ONLY, ResultSet.TYPE_SCROLL_INSENSITIVE, and ResultSet.TYPE_SCROLL_SENSITIVE. • Learn about the concurrency types of result sets: ResultSet.CONCUR_READ_ONLY and ResultSet.CONCUR_UPDATABLE.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	**1. Introduction** (5 minutes) - Briefly recap the JDBC process and the role of result sets in retrieving data. - Emphasize the importance of understanding different result set types for efficient data manipulation. **2. Development** (30 minutes) a. Explain the three main result set types: - ResultSet.TYPE_FORWARD_ONLY: Can only move forward through the result set. - ResultSet.TYPE_SCROLL_INSENSITIVE: Can move both forward and backward, but changes made by other users are not reflected. - ResultSet.TYPE_SCROLL_SENSITIVE: Can move both forward and backward, and changes made by other users are reflected. b. **Result Set Concurrency:** - Describe the two concurrency types: - ResultSet.CONCUR_READ_ONLY: Data can only be read. - ResultSet.CONCUR_UPDATABLE: Data can be modified. c. **Choosing the Right Result Set:** - Discuss factors to consider when selecting a result set type and concurrency based on application requirements. **3. Exercise (5 minutes)** - What is the default result set type in JDBC? - What is the difference between ResultSet.TYPE_SCROLL_INSENSITIVE and ResultSet.TYPE_SCROLL_SENSITIVE? - Can you update data using a result set with concurrency type ResultSet.CONCUR_READ_ONLY?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/sql/ResultSet.html	

	3. Homework - Write a Java program demonstrating the use of different result set types and concurrency modes. - Experiment with updating data using an updatable result set. - Research and understand the performance implications of different result set types..
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

<table>
<tr><td><b>Lesson Plan No. 42</b></td><td><b>Course Name:</b> Object Oriented Programming Using Java
<b>Topic:</b> Result Set Meta Data.</td><td><b>Course No.: PSCSATC-223</b></td></tr>
<tr><td><b>Objectives</b></td><td colspan="2">At the end of the lesson the student shall be able to:
• Understand the concept of Result Set Metadata.
• Learn how to obtain metadata information about a ResultSet.
• Explore the methods used to retrieve column information from Result Set Metadata.</td></tr>
<tr><td><b>Teaching Aids (if any)</b></td><td colspan="2">a. Power Point Presentation</td></tr>
<tr><td><b>Teaching Development</b></td><td colspan="2">1. <b>Introduction</b> (5 minutes)
- Briefly recap the concept of ResultSet and its role in JDBC.
- Explain the need for understanding the structure of the data returned by a ResultSet.
- Introduce the concept of Result Set Metadata as a way to obtain information about the ResultSet's columns.
2. <b>Development</b> (30 minutes)
a. <b>Obtaining Result Set Metadata:</b>
- Demonstrate how to get a ResultSetMetaData object from a ResultSet.
- Explain the purpose of ResultSetMetaData.
b. <b>Retrieving Column Information:</b>
- Discuss the methods available to get column count, column names, column types, and other metadata.
- Provide code examples to illustrate these methods.
c. <b>Using Metadata:</b>
- Explain how to use metadata information to dynamically process ResultSet data.
- Discuss potential use cases for Result Set Metadata (e.g., generating reports, populating UI components).
3. <b>Exercise (5 minutes)</b>
- What is Result Set Metadata?
- How do you obtain a ResultSetMetaData object?
- Name two methods used to retrieve column information from ResultSetMetaData.</td></tr>
<tr><td><b>Closure</b></td><td colspan="2">1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.
2. Suggested Reading
https://docs.oracle.com/javase/8/docs/api/java/sql/ResultSetMetaData.html
3. Homework
- Write a Java program to retrieve a ResultSet from a database query.
- Use ResultSetMetaData to print the column names and data types of the result set.
- Experiment with other methods of ResultSetMetaData to explore different metadata information.</td></tr>
</table>

	-
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 43	**Course Name:** Object Oriented Programming Using Java **Topic:** Catching Database Results	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand how to retrieve data from a database using JDBC. • Learn how to handle different types of ResultSet objects. • Grasp the importance of error handling and exception handling while working with databases.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Briefly recap the JDBC architecture and process covered in the previous lesson. - Emphasize the importance of fetching data from the database as the core functionality of any database application. - Introduce the concept of ResultSet and its role in storing query results. 2. **Development** (30 minutes) a. **ResultSet Interface:** - Explain the ResultSet interface and its methods for navigating and retrieving data (next(), previous(), absolute(), relative(), getXXX()). - Demonstrate how to iterate through a ResultSet to extract data. b. **ResultSet Types and Concurrency:** - Discuss the different types of ResultSet (TYPE_FORWARD_ONLY, TYPE_SCROLL_INSENSITIVE, TYPE_SCROLL_SENSITIVE) and their characteristics. - Explain the concept of concurrency (CONCUR_READ_ONLY, CONCUR_UPDATABLE) and its implications. c. **Error and Exception Handling:** - Highlight the potential exceptions that can occur during database operations (SQLException, ClassNotFoundException, etc.). - Demonstrate how to use try-catch blocks to handle exceptions gracefully. 3. **Exercise (5 minutes)** - What is a ResultSet and how is it used in JDBC? - Name two methods to navigate through a ResultSet. - What is the importance of error handling when working with databases?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.	

	2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html l 3. Homework - Write a Java program to retrieve data from a table, display it in a formatted manner, and handle potential exceptions. - Experiment with different ResultSet types and concurrency modes. - Explore advanced ResultSet features like updating data through the ResultSet.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 44	**Course Name:** Object Oriented Programming Using Java **Topic:** Handling database Queries	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the different types of SQL queries (SELECT, INSERT, UPDATE, DELETE). • Learn how to execute SQL queries using JDBC. • Grasp the concept of result sets and how to process them.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) • Briefly recap the JDBC architecture and connection establishment. • Emphasize the importance of SQL queries for interacting with databases. • Introduce the different types of SQL queries (CRUD operations). 2. **Development** (30 minutes) a. **SELECT Query:** - Explain the syntax and usage of SELECT queries. - Demonstrate how to execute SELECT queries using JDBC and process the results using ResultSet. - Discuss various clauses like WHERE, ORDER BY, GROUP BY, HAVING. b. **INSERT, UPDATE, DELETE Queries:** - Explain the syntax and usage of these DML operations. - Demonstrate how to execute these queries using JDBC. - Discuss transaction management for data integrity. c. **Prepared Statements:** - Reiterate the benefits of prepared statements for performance and security. - Demonstrate how to create and execute prepared statements for queries. 1. **Exercise (5 minutes)** - What is the primary use of a SELECT query? - How do you execute an SQL query using JDBC? - What is the purpose of a prepared statement?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://www.w3schools.com/sql/ 3. Homework - Write a Java program to perform CRUD operations on a database table using JDBC. - Experiment with different SQL clauses and functions.	

	- Implement error handling for database operations. Explore advanced ResultSet features like updating data through the ResultSet.
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 45	**Course Name:** Object Oriented Programming Using Java **Topic:** JDBC and AWT	**Course No.: PSCSATC-223**
Objectives	At the end of the lesson the student shall be able to: • Understand the basic concepts of JDBC and AWT. • Learn how to integrate JDBC and AWT to create database-driven GUI applications. • Be able to design and implement a simple database-based GUI application.	
Teaching Aids (if any)	a. Power Point Presentation	
Teaching Development	1. **Introduction** (5 minutes) - Briefly recap JDBC and its role in database connectivity. - Introduce AWT as a GUI toolkit for Java. - Explain the potential of combining JDBC and AWT to build interactive database applications. 2. **Development** (30 minutes) a. **Database Connection and Data Retrieval:** - Demonstrate how to establish a database connection using JDBC. - Explain how to execute SQL queries to retrieve data. - Show how to process the retrieved data and store it in suitable data structures. b. **AWT Components for Data Display:** - Introduce basic AWT components like labels, text fields, and text areas. - Explain how to create a simple GUI layout using AWT. - Demonstrate how to populate AWT components with data retrieved from the database. c. **Event Handling and User Interaction:** - Introduce the concept of event handling in AWT. - Explain how to handle user interactions like button clicks. - Demonstrate how to update the database based on user input. 3. **Exercise (5 minutes)** - What is the purpose of integrating JDBC and AWT? - Name two AWT components that can be used to display data from a database. - How do you typically handle user interactions in an AWT application?	
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading https://docs.oracle.com/javase/8/docs/api/java/awt/package-summary.html	

	3. Homework - Create a simple GUI application using AWT to display a list of students from a database. - Implement a search functionality to filter the students based on a given criteria
Evaluation	1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss. Spend 5 minutes to evaluate student assimilation of the lesson contents