**Java Basic Programs Topic Wise**

**Basic Input/ Outputs programs:-**

1. Find the square root of a given number, number input by the user.
2. WAP to calculate the Simple Interest.
3. How to swap two numbers without using third variable.
4. WAP to find area and perimeter of rectangle.
5. WAP to find volume and surface area of cuboid.

**Flow Controls:-**

1. WAP to find the largest of three numbers.
2. Write a Java program to calculate the Phone Bill based on the number of minutes used.

The first 50 minutes are free, the next 100 minutes cost 1 Rs. per minutes and any additional minutes cost 2 Rs. per minutes.

1. WAP to calculate the Interest Earned on a bank deposit. For, deposit up to 50,000 Rs., the interest rate is 3%,

for the next 50,000 Rs. the interest rate is 4% , for any amount above 1,00,000 Rs. the interest rate is 5%.

1. WAP to calculate the total fare of a train journey based on distance travelled. For the first 50 km,

The fare is 3 Rs/km, for the next 100 km, the fare is 2 Rs/km, for the next 150 Km the fare is 1.5 Rs/km and any distance beyond 300 km is charged at 1 Rs/km, if the total distance exceeds 500 km, provide a 10% discount on the total fare.

1. How to print all alphabet using for loop program in Java.
2. Find the greatest common devisor of two Numbers.
3. Write a program to find LCM of two numbers.
4. Write a program to find the sum of n natural numbers.
5. Write a program to convert decimal into binary.
6. Write a program to convert decimal to hex decimal.
7. WAP to check the given number is Peterson number or not.
8. WAP to find factorial of a given number.
9. WAP to check the given number is a Armstrong or not.
10. WAP to Generate Random number.
11. WAP to print Following Pattern --->

1

2 3

4 5 6

7 8 9 10

1. WAP To print the following Pattern--->

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

1. WAP to Print Fibonacci Series.

**String Manipulation:-**

1. WAP to check the given string is a string palindrome or not.
2. WAP to print repetitive character in a given string.
3. Check whether a character inserted is a vowel or consonant.
4. Write a program to reverse a string.
5. Write a program to count the number of words in a string.
6. Write a program to count the total number of vowels and consonant in a string.

**Array Manipulation:-**

1. Store and Print the 10 Elements in an array.
2. Sorting an array using quick sort.
3. WAP to perform insertion sort.
4. WAP to Perform Bubble sort.
5. WAP to print the elements of an array present on even position.
6. WAP to print the elements of an array present on odd position.
7. Write a program to find second highest numbers in a given array.
8. WAP to accept the marks of a student and find the total marks and percentage.

**Matrix Manipulation:-**

1. WAP to add Two Matrix in Java.
2. WAP to Transpose a Matrix.
3. WAP to check the given matrix is an identity matrix or not.
4. WAP to check given two matrices are identical or not.
5. WAP to multiply two matrices.

**Method In Java:-**

1. Make a method to calculate the power of a number.
2. Make a method to Subtract two matrices.
3. WAP to find factorial of given number using 'Recursion'.
4. WAP to generate Fibonacci series using 'Recursion'.

**OOPS, Class & Objects:-**

**1.Point2D class**

Implement the Point2D class. It should contain:

* two float fields : x , y
* non-arguments constructor which will set x , y fields to 0
* two-arguments constructor: float x , float y
* getter methods which will be responsible for returning x , y fields values
* getXY method which will return x , y values as two-element array
* setter methods which will be responsible for setting x , y fields values
* setXY method which will be responsible for setting x and y
* toString method which should return string in the following format: (x,y)

**Point3D class**

Using the Point2D class implement the Point3D class. It should extend the Point2D class. It should contain:

* private float field: z
* three-arguments constructor: float x , float y , float z
* getter method which will be responsible for returning the z field value
* getXYZ method which will return x , y, z values as three-element array
* setter method which will be responsible for setting the z field value
* setXYZ method which will be responsible for setting x , y, z
* toString method which should return string in the following format: (x,y,z)

Please provide an example usage of above implementation.

**2. Person class**

Implement the Person abstract class. It should contain:

* two String fields: name , address
* non-arguments constructor which will set name, address fields as empty strings
* two-arguments constructor: String name , String address
* getter methods which will be responsible for returning name , address field values
* setter methods which will be responsible for setting name , address fields values
* toString method which should return string in the following format: ?->?, where ? is the name and address value accordingly

**Student class**

Implement the Student class. It should extend the Person class. Implementation should meet the below criteria:

* three fields: type of study, year of study, study price
* three-arguments constructor: type of study, year of study, study price
* getter methods which will be responsible for returning declared fields
* setter methods which will be responsible for setting declared fields
* toString method which should return details information about a student

**Staff class**

Implement the Lecturer class. It should extend the Person class.

Implementation should meet the below criteria:

* two fields: specialization, salary
* two-arguments constructor: specialization, salary
* getter methods which will be responsible for returning declared fields
* setter methods which will be responsible for setting declared fields
* toString method which should return details information about a lecturer

Please provide an example usage of above implementation.

**3.** Implement the GeometricObject interface which should contain common behaviours definition for each sub class:

double getPerimeter()

double getArea()

**Circle class**

The Circle class should implement the GeometricObject interface and it should contain radius field. Methods from GeometricObject interface should be implemented according to the mathematic rules.

**4.Resizable interface**

The Resizable interface should declare the resize(int percent) method which will be responsible for rescaling objects which are implementing created interface.

**ResizableCircle class**

The ResizableCircle class should implement Resizable interface. The resize method from interface should reduce the radius of the circle in percentage.

Please provide an example usage of above implementation.

**Exception Exercise**

1. Create the divide method which has to divide the two numbers that are the attributes of the method. In case the second parameter of the method is 0, a non-default exception should be thrown: CannotDivideBy0Exception.

2.Book class

The Book class should include the following fields

* id
* title
* author
* year of release

NoBookFoundException

In case of lack of searched results an exception should be thrown. This exception should accept the String parameter object with information about which elements could not be found.

**Classes and interfaces – exercises**

1.Create the UserValidator class which with the validateEmails method will be responsible for validating user data such as: email, alternative email. Within the scope of the validateEmails method, please create the local Email class which will be responsible for formatting the provided email. Validation should cover the following scenarios:

* if the given email address is empty or it is null, set the value to unknown
* if the given email address does not meet the email criteria, set the value to unknown (use regular expressions)

**Polymorphism – exercises**

**1.Method Overriding**

* Define a superclass Animal with a method makeSound() that prints "Animal sound".
* Create subclasses Dog and Cat that override the makeSound() method to print "Bark" and "Meow" respectively.
* Create a method animalSound(Animal animal) in another class that takes an Animal object and calls its makeSound() method.

**2.Method Overloading**

* Create a class Calculator with multiple overloaded methods calculate(int a, int b), calculate(double a, double b), and calculate(String operation, int a, int b) to perform addition, subtraction, multiplication, and division based on the arguments.
* Demonstrate using each overloaded method to perform different calculations (e.g., calculate(5, 3), calculate(5.0, 3.0), calculate("multiply", 5, 3)).

**3.Combined Exercise**

* Extend the Animal hierarchy from Exercise 1 with additional subclasses (e.g., Bird, Cow).
* Override the makeSound() method in each subclass to specify the sound the animal makes.
* Create an Animal array with instances of different animals (dogs, cats, birds, cows).
* Loop through the array and call the makeSound() method for each animal, demonstrating polymorphic behavior.

**Package in Java**

1.Create a package named geometry with two classes: Point and Circle.

* Point class should have attributes x and y, constructors, and methods to calculate distance from another point.
* Circle class should have attributes centre (of type Point) and radius, constructors, and methods to calculate area and circumference.

2.Create a class TestGeometry outside the geometry package to test functionality.

* Import the Point and Circle classes from the geometry package.
* Create instances of Point and Circle, invoke their methods, and display results.

3. Extend the geometry package by adding a class Rectangle.

* Define attributes length and width, constructors, and methods to calculate area and perimeter.
* Ensure methods have appropriate access modifiers (public, protected, private) and understand their visibility within and outside the package.

4. Create a class TestRectangle outside the geometry package to test access modifiers.

* Import the Rectangle class from the geometry package.
* Attempt to access and manipulate attributes and invoke methods of Rectangle with different access levels (public, protected, private).

**Collections in java**

**Basics of ArrayList**

1.Create an ArrayList of integers and add numbers 1 to 5 to it.

* Display the contents of the ArrayList.
* Calculate and print the sum of all elements in the ArrayList.
* Remove even numbers from the ArrayList.
* Display the modified ArrayList.

**HashMap Operations**

2.Create a HashMap to store names (String) and ages (Integer) of people.

* Add at least 3 entries to the HashMap.
* Display all entries in the HashMap.
* Update the age of one person in the HashMap.
* Display the modified HashMap.

**LinkedList Operations**

3.Create a LinkedList of strings and add several elements to it.

* Display the contents of the LinkedList.
* Check if a specific element exists in the LinkedList.
* Remove the first and last elements from the LinkedList.
* Display the modified LinkedList.

**Web Application Development Using Java**

**Servlets:-**

1: Basic Servlet Handling

* Create a servlet named HelloServlet that responds with "Hello, World!"
* Map the servlet to handle requests at /hello.
* Enhance HelloServlet to accept a name parameter from the request and respond with "Hello, {name}!"
* If name parameter is not provided, respond with a generic greeting.

**2**: Servlets with Form Handling

* Create an HTML form login.html with fields for username and password.
* Submit the form to a servlet LoginServlet.
* Implement LoginServlet to validate the username and password.
* If valid credentials are entered, respond with "Login successful".
* If invalid, respond with "Invalid username or password".

**3**: Servlets with Session Management

* Enhance LoginServlet to store logged-in user information in session.
* Display the username on subsequent requests after successful login.
* Implement a logout mechanism to invalidate the session.

**Java Server Pages (JSP):-**

**1**: Basic JSP Usage

* Create a JSP page greeting.jsp that displays a simple greeting message.
* Use <%= %> for scriptlet to output "Hello, World!".
* Enhance greeting.jsp to accept a name parameter from the request and greet the user by name.
* If name parameter is not provided, display a generic greeting.

**2**: JSP with JavaBean Integration

* Create a JavaBean UserBean with properties username and email.
* Include appropriate getters and setters.
* Create a JSP page userProfile.jsp that displays user information using UserBean.
* Instantiate UserBean in the JSP page and display its properties.

**3**: JSP with Control Structures

* Create a JSP page evenNumbers.jsp that uses Java code to display all even numbers between 1 and 10.
* Use <% %> for scriptlet and <%= %> for output.
* Enhance evenNumbers.jsp to display even numbers in a formatted list using <c:forEach> tag from JSTL (JavaServer Pages Standard Tag Library).

**JSP & JDBC (Java Database Connectivity)**

**1**: Using JSP for Displaying Database Data

* Create a MySQL database named employees with a table employees containing fields id, name, email, and age.
* Insert a few records into the employees table.
* Create a JSP file displayEmployees.jsp to fetch and display all employees from the database.
* Use JDBC to establish a connection to the MySQL database.
* Execute a query to fetch all employees and display them in a tabular format.

**2**: CRUD Operations with JSP and JDBC

* Create a JSP form employeeForm.jsp to add or update employee records.
* Include fields for name, email, and age.
* Implement form submission to insert a new employee record into the database or update an existing one.
* Create a JSP file editEmployee.jsp to edit employee details.
* Provide a form pre-filled with employee details fetched from the database based on id.
* Implement form submission to update the employee record in the database.
* Create a JSP file deleteEmployee.jsp to delete an employee record.
* Display a confirmation message and handle deletion based on id using JDBC.

**Spring Boot Framework:-**

**1**: Creating a Basic REST API

* Create a Spring Boot project using Spring Initializr with the following dependencies:
* Web (for RESTful services)
* DevTools (optional, for development convenience)
* Create a UserController class with endpoints to perform CRUD operations on a User entity.
* Implement endpoints for creating, reading, updating, and deleting users.
* Use appropriate HTTP methods (GET, POST, PUT, DELETE) and path mappings (@GetMapping, @PostMapping, @PutMapping, @DeleteMapping).
* Test the endpoints using Postman or any REST client.
* Verify functionality by sending requests to create, retrieve, update, and delete users.

**2**: Data Access with JPA and MySQL

* Extend the previous exercise by integrating Spring Data JPA with MySQL database.
* Configure database connection properties (application.properties).
* Create a User entity with id, name, email, and other relevant fields.
* Define a UserRepository interface extending JpaRepository<User, Long> for CRUD operations.
* Modify UserController to use UserRepository for database operations.
* Update endpoints to interact with the database through UserRepository.
* Test the endpoints again to ensure data persistence and retrieval.
* Verify CRUD operations by checking database entries through MySQL client or any database tool.

**3**: Implementing Authentication and Authorization

* Extend the previous exercise by adding Spring Security dependency.
* Configure Spring Security to secure endpoints.
* Implement authentication using in-memory credentials or integrate with a database.
* Create a login endpoint to authenticate users.
* Implement JWT (JSON Web Token) based authentication or session-based authentication.
* Secure the UserController endpoints to allow access only to authenticated users.
* Implement role-based authorization to restrict access based on user roles.