Section - 1

Basic Inheritance(Single Inheritance):

1. Create a class Animal with properties like name and age. Create a subclass Dog that inherits from Animal and adds a property breed. Demonstrate the use of constructors in both the Animal and Dog classes.

```java
package workshop2;

class animal{
    String name;
    int age;

    public animal(String name, int age) {
        this.name= name;
        this.age = age;
    }
    public String toString() {
        return name+" "+age;
    }
}
class dog extends animal{
    String breed;

    public dog (String name,int age, String breed){
        super(name,age);
        this.breed=breed;
    }
    public String toString() {
        return name+" "+age+" "+breed;
    }

}

public class w2q1 {
    public static void main(String[] args) {
        animal a = new animal("lasta",19);
        dog d = new dog("lasta",19,"Chihuahua");
        System.out.println(a);
        System.out.println(d);
    }
}
```

```
lasta 19
lasta 19 Chihuahua
```

Method Overriding:

2. Write a Java program to create a class called Shape with methods called getPerimeter() and getArea(). Create a subclass called Circle that overrides the getPerimeter() and getArea() methods to calculate the area and perimeter of a circle.

```java
package workshop2;

class Shape{
    double getPerimeter() {
        return 0;
    }
    double getArea() {
        return 0;
    }
}
class Circle extends Shape{
    double radius;
    Circle (double radius){
        this.radius=radius;
    }
    @Override
    double getPerimeter(){
        double p;
        return p= 2*Math.PI*radius;
    }
    @Override
    double getArea(){
        double a;
        return a= Math.PI*radius*radius;
    }
}
```

```java
public class w2q2 {
    public static void main(String[] args) {
        Circle c = new Circle(10.0);
        System.out.println("Perimeter is :" + c.getPerimeter());
        System.out.println("Area is :" + c.getArea());
    }
}
```

```
Console
terminated> w2q2 (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (Dec 7, 2024, 11:1
Perimeter is :62.83185307179586
Area is :314.1592653589793
```

Super Keyword:

3. Extend the Animal and Dog example by adding a constructor to the Animal class that takes a name parameter. In the Dog class, use the super keyword to call the constructor of the Animal class. Create instances of Dog and demonstrate the use of the super keyword.

```java
package workshop2;

class animal1{
    String name;
    int age;

    public animal1(String name, int age) {
        this.name= name;
        this.age = age;
    }
    public animal1(String name) {
        this.name= name;
    }
    public String toString() {
        return name+" "+age;
    }
}
class dog1 extends animal1{
    String breed;

    public dog1 (String name,int age, String breed){
        super(name,age);
        this.breed=breed;
    }
    public dog1(String name) {
        super(name);
    }
    public String toString() {
        return name+" "+age+" "+breed;
    }

}
```

```java
public class w2q3 {
    public static void main(String[] args) {
        animal a = new animal("lasta",19);
        dog1 d = new dog1("lasta",19,"Chihuahua");
        dog1 d1 = new dog1 ("lasta");
        System.out.println(a);
        System.out.println(d);
        System.out.println(d1);
    }
}
```

Console

<terminated> w2q3 (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (Dec

```
lasta 19
lasta 19 Chihuahua
lasta 0 null
```

Protected Keyword:

4. Create a class Person with a protected attribute address. Extend it with a subclass Employee that adds a department attribute. Demonstrate how the protected keyword allows access to the address property in the Employee subclass.

```java
package workshop2;

class person{
    protected String address;

    public person(String address) {
        this.address=address;
    }
}

class employee extends person {
    String department;

    public employee (String address, String department) {
        super(address);
        this.department= department;
    }
    public String toString() {
        return address+" "+department;
    }
}

public class w2q4 {
public static void main(String[] args) {
    employee e = new employee("Kathmandu","HR");
    System.out.println(e);
}
}
```

```
Console ×
<terminated> w2q4 (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (Dec 7
Kathmandu HR
```

Access Modifiers and Inheritance:

5. Create class Parent with a private variable, a protected variable, and a public variable. Create a subclass Child and demonstrate how each type of variable is accessed (or not accessed) within the subclass.

```java
package workshop2;

class parent{
    protected String name;
    private int age;
    public String job;

    public parent (String name, int age,String job) {
        this.name=name;
        this.age= age;
        this.job=job;
    }
}
class child extends parent{
    public child(String name, int age, String job) {
        super(name,age,job);
    }
    @Override
    public String toString (){
        return name +" "+ age +" "+ job; // cannot access age
    }
}

public class w2q5 {
    public static void main(String[] args) {
        child c = new child("Prashna",19,"Student");
        System.out.println(c);
    }
}
```

Final Classes and Methods:

6. Create a final class FinalClass. Attempt to extend it with another class and observe the compiler error. Also, create a final method within a class and try to override it in a subclass.

```java
package workshop2;

final class FinalClass {
    final void method() {
        System.out.println("Final method");
    }
}
class class1 extends FinalClass{
    @Override
    void method() {
        System.out.println("method");
    }
}

public class w2q6 {
    public static void main(String[] args) {

    }
}

```

Method overloading

7. Create a class ,'Calculator' that should be able to perform addition operations for both integers and doubles. Implement the following steps:
   a. Create an instance of the Calculator class.
   b. Use the add method to add two integers (e.g., 5 and 8) and display the result.
   c. Call the add method with three integers (e.g., 10, 15, and 20) and display the result.
   d. Use the add method to add two doubles (e.g., 3.5 and 2.7) and display the result.
   e. Call the add method with three doubles (e.g., 1.1, 2.2, and 3.3) and display the result.

```java
package workshop2;

class calculator{
    int a;
    int b;
    int c;
    int add(int a, int b) {
        return a + b;
    }
    int add(int a,int b,int c) {
        return a + b;
    }
    double add(double a, double b) {
        return a + b;
    }
    double add(double a, double b,double c) {
        return a + b;
    }
}
public class w2q7 {
    public static void main(String[] args) {
        calculator c = new calculator();
        System.out.println(c.add(5,8));
        System.out.println(c.add(10.0,10.0));
        System.out.println(c.add(10,10,10));
        System.out.println(c.add(10.0,10.0,10.0));
    }
}

```

Console ×

<terminated> w2q7 (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.ex

```
13
20.0
20
20.0
```

Section - 2

Case Study

Program Description:
In this program you have to create a signup process. When the program starts, the user is given the following options:

1. Sign up
2. Quit Application

Output:

```
Please enter 1 for Sign up.
Please enter 2 for Quit.
```

Your program should keep running and enable multiple users to sign up until the quit application option is selected.
To start using the mobile app, users should sign up for an account. You are required to create a Java program for the signup process. The users will be asked to enter their full name, contact number, date of birth, password, and password confirmation.

1. The signup process must not be successful until:
   a. The full name is enter. The length must be greater than four.
   b. The mobile number has 10 digits starting with 0.
   c. The Password must initiate with capital alphabets followed by either one of @/& and ending with numeric.
      (For Example: John@0125 or John&25) .
   d. The password confirmation matches the initial entered password.
   e. The DOB is in the format DD/MM/YYYY.
   f. The user is at least 21 years old. The age should be calculated based on the year entered in the DOB (Only consider year).

2. If any of the above-mentioned conditions is not fulfilled; the sign-up process should fail, and a descriptive message should be displayed for the user explaining what has gone wrong and providing hints on the correct expected input. The program should keep asking the user to re-enter his details as long as one or more of the input fields are not correctly entered. If all fields are entered successfully, the program should stop asking the user to re-enter his details and display a message that the signup process has been completed successfully.

3. If any field is entered incorrectly, some examples of sample outputs are given below.
   Output 1:

```
You have entered the Date of Birth in invalid format.
Please start again.
```

   Output 2:

```
Your passwords are not matching.
Please start again.
```

4. If all of the above-mentioned conditions are successful, the user data is saved in appropriate data structure (Hint: Arrays can be used) to enable data checks during the login process in future builds.
   Output 1:

```
Please enter your full name: Sam Jahn

Please enter your mobile number (username): 0445544455

Please enter your password: John@21

Please confirm your password: John@21

Please enter your Date of Birth #DD/MM/YYYY (No space): 21/01/1984

You have successfully signed up.
Please enter 1 for Sign up.
Please enter 2 for Quit.
```

```java
package workshop2;

import java.util.ArrayList;
import java.util.Scanner;
import java.util.regex.Pattern;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;

public class w2q8 {
    static ArrayList<String> fullNames = new ArrayList<>();
    static ArrayList<String> mobileNumbers = new ArrayList<>();
    static ArrayList<String> passwords = new ArrayList<>();
    static ArrayList<String> datesOfBirth = new ArrayList<>();

    // Main method
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\nPlease enter 1 for Sign up.");
            System.out.println("Please enter 2 for Quit.");
            choice = scanner.nextInt();
            scanner.nextLine();  // Clear the buffer

            switch (choice) {
                case 1:
                    signUp(scanner);
                    break;
                case 2:
                    System.out.println("Quitting application.");
                    break;
                default:
                    System.out.println("Invalid option. Please try again.");
                    break;
            }
        } while (choice != 2);
```

```
private static void signUp(Scanner scanner) {
    String fullName, mobileNumber, password, confirmPassword, dob;

    // Full name validation
    while (true) {
        System.out.print("\nPlease enter your full name: ");
        fullName = scanner.nextLine();

        // Check if full name is more than 4 characters and only contains letters and spaces
        if (fullName.length() > 4 && fullName.matches("[a-zA-Z ]+")) {
            break;  // Name is valid, exit the loop
        } else {
            System.out.println("Full name must be greater than 4 characters and contain only letters and spaces.");
        }
    }

    // Mobile number validation
    while (true) {
        System.out.print("Please enter your mobile number (username): ");
        mobileNumber = scanner.nextLine();
        if (Pattern.matches("0\\d{9}", mobileNumber)) {
            break;  // Mobile number is valid, exit the loop
        } else {
            System.out.println("Mobile number must have 10 digits and start with 0.");
        }
    }

    // Password validation
    while (true) {
        System.out.print("Please enter your password: ");
        password = scanner.nextLine();
        if (Pattern.matches("[A-Z][a-zA-Z]*[@&]\\d+", password)) {
            break;  // Password is valid, exit the loop
        } else {
            System.out.println("Password must start with a capital letter, followed by @/& and end with digits.");
        }
    }
```

```
    while (true) {
        System.out.print("Please confirm your password: ");
        confirmPassword = scanner.nextLine();
        if (password.equals(confirmPassword)) {
            break;  // Passwords match, exit the loop
        } else {
            System.out.println("Your passwords do not match.\n Please try again.");
        }
    }

    // Date of birth validation
    while (true) {
        System.out.print("Please enter your Date of Birth (DD/MM/YYYY): ");
        dob = scanner.nextLine();
        if (isValidDate(dob)) {
            if (isValidAge(dob)) {
                break;  // DOB and age are valid, exit the loop
            } else {
                System.out.println("You must be at least 21 years old to sign up.");
            }
        } else {
            System.out.println("You have entered the Date of Birth in invalid format. Please try again.");
        }
    }

    // If all conditions are met, store the user data
    fullNames.add(fullName);
    mobileNumbers.add(mobileNumber);
    passwords.add(password);
    datesOfBirth.add(dob);

    System.out.println("You have successfully signed up.");
}
```

```java
    // Method to validate date format and return a boolean
    private static boolean isValidDate(String dob) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        try {
            LocalDate.parse(dob, formatter);
            return true;
        } catch (DateTimeParseException e) {
            return false;
        }
    }

    // Method to check if the user is at least 21 years old
    private static boolean isValidAge(String dob) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDate birthDate = LocalDate.parse(dob, formatter);
        int currentYear = LocalDate.now().getYear();
        int birthYear = birthDate.getYear();
        return (currentYear - birthYear) >= 21;
    }
}
```

Output:

```
Please enter 1 for Sign up.
Please enter 2 for Quit.
1

Please enter your full name: Prashna Shrestha
Please enter your mobile number (username): 0123846387
Please enter your password: Prashna@47
Please confirm your password: Prashna@47
Please enter your Date of Birth (DD/MM/YYYY): 10/12/2000
You have successfully signed up.

Please enter 1 for Sign up.
Please enter 2 for Quit.
```

Error:

```
Please enter your full name: John Smith
Please enter your mobile number (username): 02
Mobile number must have 10 digits and start with 0.
Please enter your mobile number (username): 1234567898
Mobile number must have 10 digits and start with 0.
Please enter your mobile number (username): 0987654321
Please enter your password: prashna
Password must start with a capital letter, followed by @/& and end with digits.
Please enter your password: Prashna@7
Please confirm your password: prashna
Your passwords do not match.
 Please try again.
Please confirm your password: Prashna@7
Please enter your Date of Birth (DD/MM/YYYY): 101010
You have entered the Date of Birth in invalid format. Please try again.
Please enter your Date of Birth (DD/MM/YYYY): 10/12/2005
You must be at least 21 years old to sign up.
Please enter your Date of Birth (DD/MM/YYYY): 10/12/2000
You have successfully signed up.

Please enter 1 for Sign up.
Please enter 2 for Quit.
2
Quitting application.
```