CDAC Mumbai PG-DAC August 24

Assignment No- 5

1) Create a base class BankAccount with methods like deposit() and withdraw(). Derive a class SavingsAccount that overrides the withdraw() method to impose a limit on the withdrawal amount. Write a program that demonstrates the use of overridden methods and proper access modifiers & return the details.

```
class BankAccount {
    private double balance;

    public BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposit successful. New balance: " + balance);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawal successful. New balance: " + balance);
        } else {
            System.out.println("Invalid withdrawal amount.");
        }
    }

    public double getBalance() {
```

CDAC MUMBAI

```
        return balance;
    }
}

class SavingsAccount extends BankAccount {
    private double withdrawalLimit;

    public SavingsAccount(double initialBalance, double withdrawalLimit) {
        super(initialBalance);
        this.withdrawalLimit = withdrawalLimit;
    }

    public void withdraw(double amount) {
        if (amount > withdrawalLimit) {
            System.out.println("Withdrawal amount exceeds the limit of " + withdrawalLimit);
        } else {
            super.withdraw(amount);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        SavingsAccount savingsAccount = new SavingsAccount(5000, 1000);

        savingsAccount.deposit(500);
        savingsAccount.withdraw(200);  // Within limit
        savingsAccount.withdraw(1500); // Exceeds limit
    }
}
```

2) Create a base class Vehicle with attributes like make and year. Provide a constructor in Vehicle to

CDAC MUMBAI

initialize these attributes. Derive a class Car that has an additional attribute model and write a constructor that initializes make, year, and model. Write a program to create a Car object and display its details.

```
class Vehicle {
    private String make;
    private int year;

    public Vehicle(String make, int year) {
        this.make = make;
        this.year = year;
    }

    public String getDetails() {
        return "Make: " + make + ", Year: " + year;
    }
}

class Car extends Vehicle {
    private String model;

    public Car(String make, int year, String model) {
        super(make, year);
        this.model = model;
    }

    public String getDetails() {
        return super.getDetails() + ", Model: " + model;
    }
}

public class Main {
    public static void main(String[] args) {
        Car car = new Car("Toyota", 2020, "Corolla");
        System.out.println(car.getDetails());
```

CDAC MUMBAI

```
    }
  }
```

3) Create a base class Animal with attributes like name, and methods like eat() and sleep(). Create a subclass Dog that inherits from Animal and has an additional method bark(). Write a program to demonstrate the use of inheritance by creating objects of Animal and Dog and calling their methods.

```
class Animal {
    protected String name;

    public Animal(String name) {
        this.name = name;
    }

    public void eat() {
        System.out.println(name + " is eating.");
    }

    public void sleep() {
        System.out.println(name + " is sleeping.");
    }
}

class Dog extends Animal {

    public Dog(String name) {
        super(name);
    }

    public void bark() {
        System.out.println(name + " is barking.");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal genericAnimal = new Animal("Generic Animal");
        genericAnimal.eat();
        genericAnimal.sleep();

        Dog dog = new Dog("Buddy");
        dog.eat();
        dog.sleep();
        dog.bark();
    }
}
```

CDAC MUMBAI

4) Build a class Student which contains details about the Student and compile and run its instance.

```
class Student {
    private String name;
    private int rollNumber;
    private double grade;

    public Student(String name, int rollNumber, double grade) {
        this.name = name;
        this.rollNumber = rollNumber;
        this.grade = grade;
    }

    public void displayDetails() {
        System.out.println("Student Name: " + name);
        System.out.println("Roll Number: " + rollNumber);
        System.out.println("Grade: " + grade);
    }

    public static void main(String[] args) {
        Student student = new Student("Alice", 101, 9.5);
        student.displayDetails();
    }
}
```

5) Write a Java program to create a base class Vehicle with methods startEngine() and stopEngine(). Create two subclasses Car and Motorcycle. Override the startEngine() and stopEngine() methods in each subclass to start and stop the engines differently.

```
class Vehicle {
    public void startEngine() {
        System.out.println("Vehicle engine started.");
    }

    public void stopEngine() {
        System.out.println("Vehicle engine stopped.");
    }
}

class Car extends Vehicle {
    @Override
    public void startEngine() {
        System.out.println("Car engine is starting with key ignition.");
    }

    @Override
    public void stopEngine() {
        System.out.println("Car engine is stopping.");
    }
}
```

CDAC MUMBAI

```
class Motorcycle extends Vehicle {
    @Override
    public void startEngine() {
        System.out.println("Motorcycle engine is starting with a kickstart.");
    }

    @Override
    public void stopEngine() {
        System.out.println("Motorcycle engine is stopping.");
    }
}

public class Main {
    public static void main(String[] args) {
        Vehicle myCar = new Car();
        myCar.startEngine();
        myCar.stopEngine();

        Vehicle myBike = new Motorcycle();
        myBike.startEngine();
        myBike.stopEngine();
    }
}
```

CDAC MUMBAI