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.

Ans. package org.example.question1;

```
class BankAccount {
    private String accountNumber;
    private double balance;

    public BankAccount(String accountNumber, double balance) {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
        } else {
            System.out.println("Insufficient balance");
        }
    }

    public double getBalance() {
        return balance;
    }


    public String getAccountNumber() {
        return accountNumber;
    }
}


class SavingsAccount extends BankAccount {
    private static final double MIN_BALANCE = 100.0;


    public SavingsAccount(String accountNumber, double balance) {
```

```
        super(accountNumber, balance);
    }

    @Override
    public void withdraw(double amount) {
        if (getBalance() - amount < MIN_BALANCE) {
            System.out.println("Minimum balance of $" + MIN_BALANCE + " required!");
        } else {
            super.withdraw(amount);
        }
    }
}

public class Program1 {

            public static void main(String[] args) {

                        BankAccount ba = new BankAccount("BA1234", 500);
                        System.out.println("BankAccount created with account number: " +
ba.getAccountNumber() + " and initial balance: $" + ba.getBalance());

        // Deposit money into the BankAccount
        ba.deposit(1000);
        System.out.println("Deposited $1000. New balance: $" + ba.getBalance());

        // Withdraw money from the BankAccount
        ba.withdraw(600);
        System.out.println("Withdrew $600. New balance: $" + ba.getBalance());

        // Create a SavingsAccount object
        SavingsAccount sa = new SavingsAccount("SA1234", 450);
        System.out.println("\nSavingsAccount created with account number: " + sa.getAccountNumber() + "
and initial balance: $" + sa.getBalance());

        // Attempt to withdraw money from the SavingsAccount
        sa.withdraw(300);
        System.out.println("Balance after attempting to withdraw $300: $" + sa.getBalance());

        // Attempt to withdraw money that would drop balance below minimum
        sa.withdraw(400);
        System.out.println("Balance after attempting to withdraw $400: $" + sa.getBalance());
    }

            }
```

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

Ans. package org.example.question2;

```
class Vehicle{
	private String make;
	private int year;
	public Vehicle(String make, int year) {
		this.make = make;
		this.year = year;
	}
	public String getMake() {
		return make;
	}
	public int getYear() {
		return year;
	}

	public void displayDetails() {
		System.out.println("Make: " +make);
		System.out.println("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 getModel() {
		return model;
	}
	 @Override
	   public void displayDetails() {
	      super.displayDetails(); // Call the superclass method
	      System.out.println("Model: " + model);
	   }
}

public class Program2 {

	public static void main(String[] args) {
		Car car = new Car("Toyota", 2022, "fortuner");
		System.out.println("Car details : ");
		car.displayDetails();
```

```
	}

}
```

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.

Ans.
```
package org.example.question3;
class Animal{
	private 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 ");
	}

	public String getName() {
		return name;
	}
}

class Dog extends Animal{

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

	public void bark() {
		System.out.println(getName() + " is barking ");
	}
}
public class Program3 {

	public static void main(String[] args) {
		Animal animal = new Animal("Pet Animal");
		System.out.println("Animal Details : ");
		animal.eat();
		animal.sleep();
```

```
		Dog dog = new Dog("oreo");
		System.out.println();
		System.out.println("Dpg details :" );
		dog.eat();
		dog.sleep();
		dog.bark();
	}

}
`
```

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

Ans. package org.example.question4;

```
class student{
private String name;

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

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

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

	public String getName() {
		return name;
	}
}

class completeInfo extends student{

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

	public void classType() {
		System.out.println(getName() + " is in high School ");
	}

}
public class Program4 {
```

```
	public static void main(String[] args) {
		student stu = new student("rohan");
		System.out.println("Student details : ");
		stu.studying();
		stu.learning();

		completeInfo cominfo = new completeInfo("rohan");
		System.out.println("Student complete details : ");
		cominfo.classType();
	}

}
```

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.

Ans.

```
package org.example.question5;


abstract class Vehicle{

	public abstract void startEngine();
	public abstract void stopEngine();

}

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

	@Override
	public void stopEngine() {
		System.out.println("Car engine stopped when the key was turned off.");
	}
}
class Motorcycle extends Vehicle{
	@Override
	public void startEngine() {
		System.out.println("Motorcycle engine started with a kick-start.");
	}

	@Override
	public void stopEngine() {
		System.out.println("Motorcycle engine stopped when the ignition was turned off.");
	}
}
```

```
public class Program5 {

	public static void main(String[] args) {
	Vehicle car = new Car();
	Vehicle motorcycle = new Motorcycle();

	startAndStopEngine(car);
	startAndStopEngine(motorcycle);

	}

	public static void startAndStopEngine(Vehicle vehicle) {
		vehicle.startEngine();
	   vehicle.stopEngine();
	   }

}
```