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 – `BankAccount.java`

```
package cdac.inherit_bankAccount_q1;

public class BankAccount {

    private String accountName;
    private int accountNumber;
    private double balance;

    public BankAccount(String accountName, int accountNumber, double balance) { // constructor
        this.accountName = accountName;
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    public double getBalance() { // getter
        return balance;
    }

    public void setBalance(double balance) { // setter
        this.balance = balance;
    }

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

```
    }

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

    public String getAccountDetails() {
        return "Account Name : " + accountName + ", Account Number : " +
    accountNumber + ", Account balance : "
                + balance;
    }
}
```

SavingAccount.java

```
package cdac.inherit_savingAccount_q1;

import cdac.inherit_bankAccount_q1.BankAccount;

public class SavingAccount extends BankAccount {
    private double withdrwalLimit;

    public SavingAccount(String accountName, int accountNumber, double balance,
    double withdrwalLimit) {
        super(accountName, accountNumber, balance); // using super keywords,
    fields from parent class is broght in child

        // class
        this.withdrwalLimit = withdrwalLimit;
    }
```

```
    @Override
    public void withdraw(double amount) {

        if (amount >= withdrwalLimit) {
            System.out.println(withdrwalLimit + ", Withdrawal amount here, is
    exceeding limit");
        } else {
            super.withdraw(amount); // using super called withdraw method of
    parent-class
        }

    }

    @Override
    public String getAccountDetails() {
        return super.getAccountDetails() + ", Withdraw Limit : " + withdrwalLimit;
    }

}
```

Program.java

```
package cdac.inherit_main_q1;

import cdac.inherit_savingAccount_q1.SavingAccount;
import cdac.inherit_bankAccount_q1.BankAccount;

public class Program {

    public static void main(String[] args) {
        BankAccount b = new BankAccount("Rupali", 12345, 60000);
        System.out.println(b.getAccountDetails());
        b.deposit(5000);
        b.withdraw(3000);
        System.out.println(b.getAccountDetails());

        SavingAccount s = new SavingAccount("Rupali", 12345, 50000, 4000);
        System.out.println(s.getAccountDetails());
```

```
            s.withdraw(3000);
            s.withdraw(6000);
            System.out.println(s.getAccountDetails());

        }

}
```

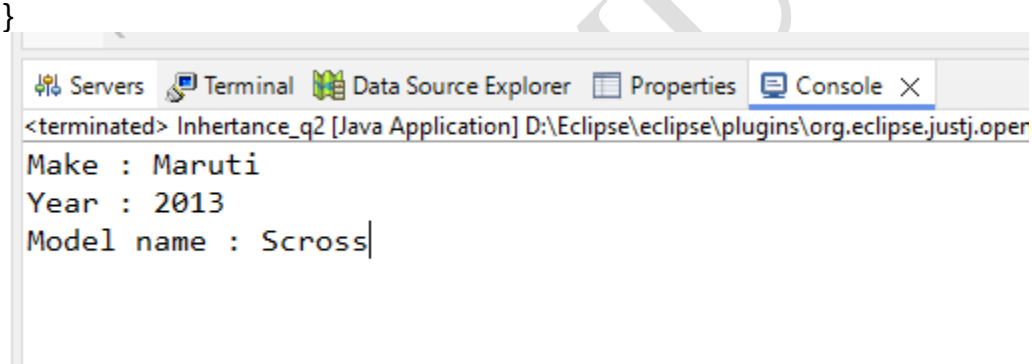
```
Servers  Terminal  Data Source Explorer  Properties  Console
<terminated> Program (5) [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (17
Account Name : Rupali, Account Number : 12345, Account balance : 60000.0
Account Name : Rupali, Account Number : 12345, Account balance : 62000.0
Account Name : Rupali, Account Number : 12345, Account balance : 50000.0, Withdraw Limit : 4000.0
4000.0, Withdrawal amount here, is exceeding limit
Account Name : Rupali, Account Number : 12345, Account balance : 47000.0, Withdraw Limit : 4000.0
```

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 - `Inhertance_q2.java`

```
package cdac.inheritance_q2;

class Vehicles {
        private String make;
        private int year;

        public Vehicles(String make, int year) { // constructor
                this.make = make;
                this.year = year;
        }

        public String getMake() { // getter
                return make;
        }

        public int getYear() { // getter
                return year;
        }

}

class Car extends Vehicles {
        private String model;
```

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

	public String getCarDetails() {
		return "Make : " + getMake() + "\n" + "Year : " + getYear() + "\n" + "Model
name : " + model;

	}
}

public class Inhertance_q2 {

	public static void main(String[] args) {
		Car stu = new Car("Maruti", 2013, "Scross");
		System.out.println(stu.getCarDetails());

	}

}
```



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 - `Inheritance_q3.java`

```
package cdac.inheritance_q3;

class Animal {

	private String name;

	public Animal(String name) {
```

```
		this.name = name;
	}

	public void eat() {
		System.out.println("In eat method");
	}

	public void sleep() {
		System.out.println("In sleep method");
	}
}

class Dog extends Animal {

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

	public void eat() {
		super.eat();
		System.out.println("In dog's eat method");
	}

	public void sleep() {
		super.sleep();
		System.out.println("In dog's sleep method");
	}

	public void bark() {
		System.out.println("In bark method");
	}
}

public class Inheritance_q3 {

	public static void main(String[] args) {
		Animal a = new Animal("cat");
		a.eat();
		a.sleep();

		Dog d = new Dog("Tomy");
		d.eat();
		d.sleep();
		d.bark();
	}
```

}

```
Servers  Terminal  Data Source Explorer  Properties  Console
<terminated> Inheritance_q3 [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.jus
In eat method
In sleep method
In eat method
In dog's eat method
In sleep method
In dog's sleep method
In bark method
```

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

Ans - `Inhertance_q4.java`

```
package cdac.inheritance_q4;

class student {
	String name;
	int classNumber;
	int rollNum;

	public student() {
		this.name = "Rupali";
		this.classNumber = 12;
		this.rollNum = 31;
	}

	public String getStudentDetails() {
		return "Name : " + name + "\n" + "Class : " + classNumber + "\n" +
"Roll Number : " + rollNum;

	}
}

public class Inhertance_q4 {

	public static void main(String[] args) {
		student stu = new student();
//		stu.getStudentDetails();
		System.out.println(stu.getStudentDetails());

	}
```

```
}
```

```
Console
<terminated> Inhertance_q4 [Java Application] D:\Eclipse\eclipse\plugins\or
Name : Rupali
Class : 12
Roll Number : 31
```

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 - `Inheritance_q5.java`

```
package cdac.inheritance_q5;

abstract class Vehicle {

	abstract void startEngine();

	abstract void stopEngine();
}

class Car extends Vehicle {

	public void startEngine() {
		System.out.println("In car startEngine method");
	}

	public void stopEngine() {
		System.out.println("In car stopEngine method");
	}

}

class Motorcycle extends Vehicle {

	public void startEngine() {
		System.out.println("In motorcycle startEngine method");
	}

	public void stopEngine() {
		System.out.println("In motorcycle stopEngine method");
```

```
        }
}

public class Inheritance_q5 {

        public static void main(String[] args) {
                Car c = new Car();
                c.startEngine();
                c.stopEngine();

                Motorcycle m = new Motorcycle();
                m.startEngine();
                m.stopEngine();
        }

}
```