Name : Shreyash Shete
Batch : May 2022

Core Java Assignments

Inheritance and Polymorphism

Objective: At the end of the assignments, participants will be able to create abstract classes, create a new class by extending an existing class, write code to exhibit polymorphic behavior of a method call, use of interfaces

Concept: Inheritance and Polymorphism

Estimated Time : 2.5 Hours

1. Create an abstract class Instrument which is having the abstract function play.
   Create three more sub classes from Instrument which is Piano, Flute, Guitar.
   Override the play method inside all three classes printing a message

   "Piano is playing tan tan tan tan " for Piano class
   "Flute is playing toot toot toot toot" for Flute class
   "Guitar is playing tin tin tin " for Guitar class

   You must not allow the user to declare an object of Instrument class.

   Create an array of 10 Instruments.

   Assign different type of instrument to Instrument reference.

   Check for the polymorphic behavior of play method.

   Use the instanceof operator to print that which object stored at which index of instrument array.

Class Instrument :

```
public abstract class Instrument {

	public abstract void play();

}
```

Class Flute :

```
public class Flute extends Instrument {

	public Flute() {

	}

	public void play() {
		System.out.println("Flute is playing toot toot toot
toot");
	}
}
```

Class Guitar :

```
public class Guitar extends Instrument {

	public Guitar() {

	}

	public void play() {
		System.out.println("Guitar is playing tin tin tin..");
	}

}
```

Class Piano :

```
public class Piano extends Instrument {

	public Piano() {

	}

	public void play() {
		System.out.println("The piano is playing tan tan
tan..");
	}
}
```

Class InstrumentMain :

```
import java.util.Scanner;

public class InstrumentMain {

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		int ch = 0;

		do {

			System.out.println("How many instruments do you
want : ");
			int n = sc.nextInt();

			Instrument obj[] = new Instrument[10];

			obj[0] = new Flute();
			obj[1] = new Flute();
			obj[2] = new Flute();
```

```
                obj[3] = new Flute();

                obj[4] = new Guitar();
                obj[5] = new Guitar();
                obj[6] = new Guitar();

                obj[7] = new Piano();
                obj[8] = new Piano();
                obj[9] = new Piano();

                display(obj);

                System.out.println("Do you want to continue ?
Press 1 ");
                ch = sc.nextInt();

           }while(ch==1);
           System.out.println("---Thank you----");
     }


     public static void display(Instrument arr[]) {

           for(int i=0;i<arr.length;i++) {
                if(arr[i] instanceof Piano) {
                     System.out.println("Piano is present in the
instrument array at position : "+ arr[i] + 1);
                     arr[i].play();
                }
                else if(arr[i] instanceof Guitar) {
                     System.out.println("Guitar is present in
the instrument array at position : " + arr[i] + 1);
                     arr[i].play();
                }
                else if(arr[i] instanceof Flute) {
                     System.out.println("Flute is present in the
instrument array at position : " + arr[i] + 1);
                     arr[i].play();

                }
           }
     }
}
```

2. Create an abstract class Compartment to represent a rail coach. Provide a abstract function notice in this class. Derive FirstClass, Ladies, General, Luggage classes from the compartment class. Override the notice function in each of them to print notice suitable to the type of the compartment.

   Create a class TestCompartment . Write main function to do the following:

   Declare an array of Compartment pointers of size 10.

   Create a compartment of a type as decided by a randomly generated integer in the range 1 to 4.

   Check the polymorphic behavior of the notice method.

Class Compartment :

```
public abstract class Compartment {

	public abstract void contents();

	}
```

Class FirstClass :

```
public class FirstClass extends Compartment {

	public FirstClass() {

	}

	public void contents() {
		System.out.println("---This is First Class
Compartment---");
		System.out.println("\n");
	}
```

```
}
```

Class Ladies :

```
public class Ladies extends Compartment {

	public Ladies() {

	}

	public void contents() {
		System.out.println("---This is Ladies Compartment---
");
		System.out.println("\n");
	}

}
```

Class Luggage :

```
public class Luggage extends Compartment {

	public Luggage() {

	}

	public void contents() {
		System.out.println("---This is Luggage Compartment---
");
		System.out.println("\n");
	}

}
```

Class TestCompartment :

```
import java.util.Scanner;

public class TestCompartment {

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		int ch = 0;

		do {

			System.out.println("How many Compartments do you
want : ");
			int n = sc.nextInt();

			Compartment obj[] = new Compartment[10];

			obj[0] = new FirstClass();
			obj[1] = new FirstClass();
			obj[2] = new FirstClass();

			obj[3] = new General();
			obj[4] = new General();

			obj[5] = new Ladies();
			obj[6] = new Ladies();
			obj[7] = new Ladies();

			obj[8] = new Luggage();
			obj[9] = new Luggage();

			display(obj);

			System.out.println("Do you want to continue ?
Press 1 ");
			ch = sc.nextInt();

		}while(ch==1);
		System.out.println("-----Thank you, Visit again ! ");
	}
```

```
	public static void display(Compartment arr[]) {

		for(int i=0;i<arr.length;i++) {
			if(arr[i] instanceof FirstClass) {
				System.out.println("FirstClass Compartment
is present in the train !");
				arr[i].contents();
			}
			else if(arr[i] instanceof General) {
				System.out.println("General Compartment is
present in the train ! ");
				arr[i].contents();
			}
			else if(arr[i] instanceof Ladies) {
				System.out.println("Ladies Compartment is
present in the train ! ");

				arr[i].contents();
			}
			else if(arr[i] instanceof Luggage) {
				System.out.println("Luggage Compartment is
present in the train ! ");
				arr[i].contents();
			}
		}
	}
}
```

3. Create a class Medicine to represent a drug manufactured by a pharmaceutical company. Provide a function displayLabel() in this class to print Name and address of the company.

   Derive Tablet, Syrup and Ointment classes from the Medicine class. Override the displayLabel() function in each of these classes to print additional information suitable to the type of medicine. For example, in case of tablets, it could be “store in a cool dry place”, in case of ointments it could be “for external use only” etc.

   Create a class TestMedicine . Write main function to do the following:
   Declare an array of Medicine references of size 10
   Create a medicine object of the type as decided by a randomly generated integer in the range 1 to 3.

   Refer Java API Documentation to find out random generation feature.

   Check the polymorphic behavior of the displayLabel() method.

Class Medicine :

```
public abstract class Medicine {

	public abstract void displayLabel();

}
```

Class Tablet :

```
public class Tablet extends Medicine {

	public Tablet() {
```

```
	}

	public void displayLabel() {
		System.out.println("Store in cool and Dry place ! ");
		System.out.println("\n");
	}

}
```

Class Syrup :

```
public class Syrup extends Medicine {

	public Syrup() {

	}

	public void displayLabel() {
		System.out.println("Store in cool and dry place ! ");
		System.out.println("\n");
	}

}
```

Class Ointment :

```
public class Ointment extends Medicine {

	public Ointment() {

	}

	public void displayLabel() {
		System.out.println("For External Use only !!");
		System.out.println("\n");
	}

}
```

Class TestMedicine :

```
import java.util.Scanner;

public class TestMedicine {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int ch=0;

		do {

			System.out.println("How many Medicines do you
want : ");
			int n = sc.nextInt();

			Medicine obj[] = new Medicine[10];

			obj[0] = new Tablet();
			obj[1] = new Tablet();
			obj[2] = new Tablet();

			obj[3] = new Syrup();
			obj[4] = new Syrup();
			obj[5] = new Syrup();

			obj[6] = new Ointment();
			obj[7] = new Ointment();
			obj[8] = new Ointment();
			obj[9] = new Ointment();

			display(obj);

			System.out.println("Do you want to continue ?
Press 1");
			ch = sc.nextInt();

		}while(ch==1);
		System.out.println("----Thank you for our Service----
");
	}


	public static void display(Medicine arr[] ) {
```

```
            for(int i=0;i<arr.length;i++) {
                  if(arr[i] instanceof Tablet) {
                        System.out.println("The Medicine 'Tablet'
is present in the Store ");
                        arr[i].displayLabel();
                  }
                  else if(arr[i] instanceof Syrup) {
                        System.out.println("The Medicine 'Syrup' is
present in the Store ");
                        arr[i].displayLabel();
                  }
                  else if(arr[i] instanceof Ointment) {
                        System.out.println("The Medicine 'Ointment'
is present in the Store ");
                        arr[i].displayLabel();
                  }
            }
      }
}
```

4. Write a program that accepts two numbers and a operator like (+,-,*, /) as command line arguments and perform the appropriate operation indicated by operator.

   If the user enters any other character the appropriate message will be displayed. The output of the program should be displayed to the user.

Class Inheritance4 :

```
public class Inheritance4 {

	public static void main(String[] args) {

		System.out.println("\t---Program for Command line
argument for various operators..");
		if(args.length==0)
		{
			System.out.println("No arguments are passed");
		}
		else
		{
			int a=Integer.parseInt(args[0]);
			String p=args[1];
			int b=Integer.parseInt(args[2]);

			switch(p)
			{
			case "+":
				System.out.println("Addition of "+a+" and
"+b+" : "+(a+b));
				break;

			case "-":
				System.out.println("Subtraction of "+a+"
and "+b+" : "+(a-b));
				break;
```

```
                case "*":
                        System.out.println("Multiplication of "+a+"
and "+b+" : "+(a*b));
                        break;

                case "/":
                        System.out.println("Division of "+a+" and
"+b+" : "+(a/b));
                        break;

                case "%":
                        System.out.println("Modulo of "+a+" and
"+b+" : "+(a%b));
                        break;


                default:
                        System.out.println("Please Enter '+', '-',
'*', '/' & '%' operator only.");

                }
            }
        }
}
```

5. Create a class Car which contains members speed, noOfGear. The class has a method drive() which is responsible to provide starting speed and noOfGears to a Car. Implement display() method which will display all attributes of Car class.

   The class SportCar is derived from the class Car which adds new features AirBallonType. When this method is invoked, initial speed and gear status must be displayed on console. Override the display method which display all attribute of the SportCar. Make use of super class display() method.

Class Car :

```
import java.util.Scanner;

public abstract class Car {
	Scanner sc = new Scanner(System.in);

	public int speed, noOfGear;

	public void drive() {

		System.out.println("Enter Speed and Gear of the Car :
");
		speed = sc.nextInt();
		noOfGear = sc.nextInt();
	}

	public void display() {
		System.out.println("Speed is : " + speed);
		System.out.println("No of Gears are : " + noOfGear);
	}

}
```

Class SportCar :

```
public class SportCar extends Car {

	public String s = "Air balloon type";

	public void display() {
		super.display();
		System.out.println("Special Feature is : " + s);
	}

}
```

Class Inheritance5 :

```
public class Inheritance5 {

	public static void main(String[] args) {

		SportCar sc = new SportCar();
		sc.drive();
		sc.display();

	}

}
```