

PROGRAM-1

Aim: Write a program to implement parameterized constructor in Java.

Create a class Box that uses a parameterized constructor to initialize the dimensions of a box. The dimensions of the Box are width, height, depth. The class should have a method that can return the volume of the box. Create an object of the Box class and test the functionalities.

Overview: A constructor that has parameters is known as parameterized constructor. If we want to initialize fields of the class with our own values, then use a parameterized constructor.

Syntax of Parameterized Constructor in Java:

```
class ClassName {
    TypeName variable1;
    TypeName variable2;
    ClassName(TypeName variable1, TypeName variable2) {
        this.variable1 = variable1;
        this.variable2 = variable2;
    }
}
```

Source Code:

```
package Java;
import java.util.Scanner;

class Box {
    int width,height,depth;
    Box (int w, int h, int d) {
        width=w;
        height=h;
        depth=d;
    }
    int calVolume() {
        return width*height*depth;
    }
}

public class Program1 {
    public static void main(String[] args) {
        System.out.println("Parameterized Constructor");
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the dimensions: ");
        int w=s.nextInt();
        int h=s.nextInt();
        int d=s.nextInt();
        Box b = new Box(w,h,d);
        System.out.println("The volume of the box is: "+ b.calVolume());
    }
}
```

Output:

```
Parameterized Constructor  
Enter the dimensions: 23 12 9  
The volume of the box is: 2484
```

PROGRAM-2

Aim: Write a program to implement method overriding in Java.

Create a base class Fruit which has name ,taste and size as its attributes. A method called eat() is created which describes the name of the fruit and its taste. Inherit the same in 2 other class Apple and Orange and override the eat() method to represent each fruit taste.

Overview: Method overriding is a fundamental concept in object-oriented programming (OOP) that allows a subclass (child class) to provide a specific implementation for a method inherited from its superclass (parent class). This enables customization of behavior based on the specific subclass without modifying the superclass itself.

Syntax of Method Overriding in Java:

```
class ClassName1 {
    TypeName variable1;
    TypeName variable2;
    returnType method (parameters) {}
}
class ClassName2 extends ClassName1 {
    returnType method (parameters) {}
}
```

Source Code:

```
package Java;

class Fruit {
    String name,taste;
    int size;
    void eat() {
        System.out.println("Inside Fruit class");
    }
}

class Apple extends Fruit {
    Apple (String n, String t, int s) {
        name=n;
        taste=t;
        size=s;
    }
    void eat() {
        System.out.println("Name: "+ name +"\nTaste: "+ taste);
    }
}

class Orange extends Fruit {
    Orange (String n, String t, int s) {
        name=n;
        taste=t;
        size=s;
    }
}
```

```
}  
void eat() {  
System.out.println("Name: "+ name +"\nTaste: "+ taste);  
}  
}  
  
public class Program2 {  
public static void main(String[] args) {  
Apple a = new Apple("Apple","sour",10);  
System.out.println("Method Overriding");  
a.eat();  
}  
}
```

Output:

```
Method Overriding  
Name: Apple  
Taste: sour
```

PROGRAM-3

Aim: Write a program to implement polymorphism in Java.

Create a class named shape. It should contain 2 methods- draw() and erase() which should print “Drawing Shape” and “Erasing Shape” respectively. For this class we have three sub classes- Circle, Triangle and Square and each class override the parent class functions- draw () and erase (). The draw() method should print “Drawing Circle”, “Drawing Triangle”, “Drawing Square” respectively. The erase() method should print “Erasing Circle”, “Erasing Triangle”, “Erasing Square” respectively. Create objects of Circle, Triangle and Square in the following way and observe the polymorphic nature of the class by calling draw() and erase() method using each object. Shape c=new Circle(); Shape t=new Triangle(); Shape s=new Square();

Overview: Polymorphism in Java is a concept by which we can perform a single action in different ways. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms. There are two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism.

Syntax of polymorphism in Java is:

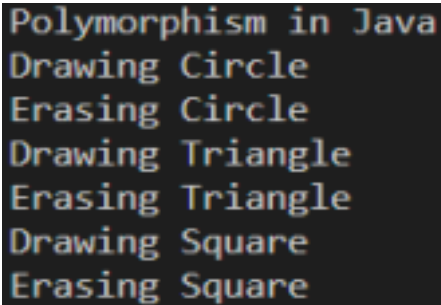
```
class ClassName1 {}  
class ClassName2 extends ClassName1 {}  
ClassName1 cn1 = new ClassName2();
```

Source Code:

```
package Java;  
  
class Shape {  
    void draw() {  
        System.out.println("Drawing Shape");  
    }  
    void erase() {  
        System.out.println("Erasing Shape");  
    }  
}  
class Circle extends Shape {  
    void draw() {  
        System.out.println("Drawing Circle");  
    }  
    void erase() {  
        System.out.println("Erasing Circle");  
    }  
}  
class Triangle extends Shape {  
    void draw() {  
        System.out.println("Drawing Triangle");  
    }  
    void erase() {  
        System.out.println("Erasing Triangle");  
    }  
}
```

```
}  
}  
class Square extends Shape {  
    void draw() {  
        System.out.println("Drawing Square");  
    }  
    void erase() {  
        System.out.println("Erasing Square");  
    }  
}  
  
public class Program3 {  
    public static void main(String[] args) {  
        Shape c = new Circle();  
        Shape t = new Triangle();  
        Shape s = new Square();  
        c.draw(); c.erase();  
        t.draw(); t.erase();  
        s.draw(); s.erase();  
    }  
}
```

Output:

A screenshot of a terminal window with a black background and light blue/grey text. The output shows the results of the Java program: 'Polymorphism in Java' followed by 'Drawing Circle', 'Erasing Circle', 'Drawing Triangle', 'Erasing Triangle', 'Drawing Square', and 'Erasing Square' on separate lines.

```
Polymorphism in Java  
Drawing Circle  
Erasing Circle  
Drawing Triangle  
Erasing Triangle  
Drawing Square  
Erasing Square
```

PROGRAM-4

Aim: Write a program to implement Exception handling in Java.

Write a Program to take care of Number Format Exception if user enters values other than integer for calculating average marks of 2 students. The name of the students and marks in 3 subjects are taken from the user while executing the program. In the same Program write your own Exception classes to take care of Negative values and values out of range (i.e. other than in the range of 0-100).

Overview: An exception is an issue (run time error) that occurred during the execution of a program. When an exception occurred the program gets terminated abruptly and, the code past the line that generated the exception never gets executed. Java exceptions cover almost all the general types of exceptions that may occur in the programming. However, we sometimes need to create custom exceptions.

Syntax of Custom Exception in Java is:

```
public class CustomException extends Exception {  
    public CustomException(String errorMessage) {  
        super(errorMessage);  
    }  
}
```

Source Code:

```
package Java;  
import java.util.Scanner;  
  
class ExceptionClass extends Exception {  
    public String toString() {  
        return "Error! Marks are not in the range(1,100)";  
    }  
}  
  
public class Program4 {  
    Scanner s = new Scanner(System.in);  
    String name;  
    float sum=0;  
    int marks[]=new int[3];  
    void input() throws ExceptionClass {  
        System.out.print("Enter name of the Student: ");  
        name=s.nextLine();  
        System.out.print("Enter the marks of "+ name +" in Physics, Chemistry and Maths: ");  
        for (int i=0;i<3; i++) {  
            marks[i]=Integer.parseInt(s.next());  
            if (marks[i]<0 || marks[i]>100) {  
                throw new ExceptionClass();  
            }  
            sum+=marks[i];  
        }  
        System.out.println("Average of marks of "+ name +" is "+ sum/3);  
    }  
}
```

```

public static void main(String[] args) {
    System.out.println("NumberFormatException, Custom Exception in Java");
    Program4 a=new Program4();
    Program4 b=new Program4();
    try {
        a.input();
        b.input();
    }
    catch (NumberFormatException n) {
        System.out.println("NumberFormatException caught "+ n.getMessage());
    }
    catch (ExceptionClass e) {
        System.out.println(e);
    }
}
}
}

```

Output:

```

NumberFormatException, Custom Exception in Java
Enter name of the Student: Abhishek
Enter the marks of Abhishek in Physics, Chemistry and Maths: 88 91 90
Average of marks of Abhishek is 89.666664
Enter name of the Student: Varun
Enter the marks of Varun in Physics, Chemistry and Maths: 90 94 78
Average of marks of Varun is 87.333336

```

```

NumberFormatException, Custom Exception in Java
Enter name of the Student: Abhishek
Enter the marks of Abhishek in Physics, Chemistry and Maths: 88 91 sw
NumberFormatException caught For input string: "sw"

```

```

NumberFormatException, Custom Exception in Java
Enter name of the Student: Abhishek
Enter the marks of Abhishek in Physics, Chemistry and Maths: 88 91 -2
Error! Marks are not in the range(1,100)

```


PROGRAM-5

Aim: Write a program to implement Exception handling in Java.

Write a program that takes as input the size of the array and the elements in the array. The program then asks the user to enter a particular index and prints the element at that index. Index starts from zero. This program may generate Array Index Out Of Bounds Exception or Number Format Exception. Use Exception handling mechanisms to handle this exception.

Overview: An exception is an issue (run time error) that occurred during the execution of a program. When an exception occurred the program gets terminated abruptly and, the code past the line that generated the exception never gets executed. The `ArrayIndexOutOfBoundsException` occurs whenever we are trying to access any item of an array at an index which is not present in the array. In other words, the index may be negative or exceed the size of an array. The `NumberFormatException` is thrown when we try to convert a string into a numeric value such as float or integer, but the format of the input string is not appropriate or illegal.

Syntax to handle any Exception in Java is:

```
try {  
    // Block of code to try  
}  
catch (Exception e) {  
    // Block of code to handle errors  
}
```

Source Code:

```
import java.util.Scanner;  
  
public class Program5 {  
    public static void main(String[] args) {  
        System.out.println("ArrayIndexOutOfBoundsException, NumberFormatException in Java");  
        try  
        {  
            Scanner s = new Scanner(System.in);  
            System.out.print("Enter the number of elements in the array: ");  
            int n=Integer.parseInt(s.next());  
            int arr[]=new int[n];  
            System.out.print("Enter the elements of the array: ");  
            for (int i=0;i<n;i++) {  
                arr[i]=Integer.parseInt(s.next());  
            }  
            System.out.print("Enter the index at which element is to be found: ");  
            int a=Integer.parseInt(s.next());  
            System.out.println("Element at index "+ a +" is "+ arr[a]);  
        }  
        catch (ArrayIndexOutOfBoundsException a) {  
            System.out.println("ArrayIndexOutOfBoundsException caught "+ a.getMessage()); }  
        catch (NumberFormatException n) {  
            System.out.println("NumberFormatException caught "+ n.getMessage());  
        }  
    }  
}
```

```
}  
}  
}
```

Output:

```
ArrayIndexOutOfBoundsException, NumberFormatException in Java  
Enter the number of elements in the array: 4  
Enter the elements of the array: 2 1 4 10  
Enter the index at which element is to be found: 3  
Element at index 3 is 10
```

```
ArrayIndexOutOfBoundsException, NumberFormatException in Java  
Enter the number of elements in the array: 4  
Enter the elements of the array: 2 1 4 10  
Enter the index at which element is to be found: 5  
ArrayIndexOutOfBoundsException caught Index 5 out of bounds for length 4
```

```
ArrayIndexOutOfBoundsException, NumberFormatException in Java  
Enter the number of elements in the array: 4  
Enter the elements of the array: 2 1 sw 10  
NumberFormatException caught For input string: "sw"
```

PROGRAM-6

Aim: Write a program to implement Socket Programming in Java.

Overview-Socket programming empowers us to establish two-way communication channels between applications running on different hosts over a network. It enables low-level control over network interactions, offering flexibility but also requiring careful handling of complexities.

Source code-

```
// A Java program for a Client
import java.io.*;
import java.net.*;

public class Client {
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;
    public Client(String address, int port)
    {
        try {
            socket = new Socket(address, port);
            System.out.println("Connected");
            input = new DataInputStream(System.in);
            out = new DataOutputStream(
                socket.getOutputStream());
        }
        catch (UnknownHostException u) {
            System.out.println(u);
            return;
        }
        catch (IOException i) {
            System.out.println(i);
            return;
        }
        String line = "";
        while (!line.equals("Over")) {
            try {
                line = input.readLine();
                out.writeUTF(line);
            }
            catch (IOException i) {
                System.out.println(i);
            }
        }
        try {
            input.close();
            out.close();
            socket.close();
        }
        catch (IOException i) {
            System.out.println(i);
        }
    }
}
```

```

    public static void main(String args[])
    {
        Client client = new Client("127.0.0.1", 5000);
    }
}

```

Server side code-

// A Java program for a Server

```
import java.net.*;
```

```
import java.io.*;
```

```
public class Server
```

```

{
    private Socket      socket = null;
    private ServerSocket server = null;
    private DataInputStream in    = null;
    public Server(int port)
    {
        try
        {
            server = new ServerSocket(port);
            System.out.println("Server started");

            System.out.println("Waiting for a client ...");

            socket = server.accept();
            System.out.println("Client accepted");
            in = new DataInputStream(
                new BufferedInputStream(socket.getInputStream()));

            String line = "";
            while (!line.equals("Over"))
            {
                try
                {
                    line = in.readUTF();
                    System.out.println(line);
                }
                catch(IOException i)
                {
                    System.out.println(i);
                }
            }
            System.out.println("Closing connection");
            socket.close();
            in.close();
        }
        catch(IOException i)
        {
            System.out.println(i);
        }
    }
}

public static void main(String args[]){Server server = new Server(5000);}

```

OUTPUT-

```
Hello  
I made my first socket connection  
Over  
Closing connection
```