2. Demonstrate the working of interface in JAVA.

```java
// Define an interface
interface Shape {
   double calculateArea();
}

// Implement the interface in a class
class Circle implements Shape {
   private double radius;

   public Circle(double radius) {
      this.radius = radius;
   }

   @Override
   public double calculateArea() {
      return Math.PI * radius * radius;
   }
}

// Another class implementing the interface
class Rectangle implements Shape {
   private double length;
   private double width;
```

```
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    @Override
    public double calculateArea() {
        return length * width;
    }
}

// Main class to test the interface implementation
public class InterfaceExample {
    public static void main(String[] args) {
        // Create objects of classes implementing the interface
        Circle circle = new Circle(5.0);
        Rectangle rectangle = new Rectangle(4.0, 6.0);

        // Call the method defined in the interface
        System.out.println("Area of Circle: " + circle.calculateArea());
        System.out.println("Area of Rectangle: " + rectangle.calculateArea());
    }
}
```

In this example, the Shape interface declares a method calculateArea(). The Circle and Rectangle classes implement this interface and provide their own implementations for calculating the area. In the main method, we create objects of these classes and call the calculateArea() method to demonstrate the use of the interface in Java.

6. What is Thread? Explain the two ways of creating a Thread in Java

Extending the Thread class: This involves creating a new class that extends the java.lang.Thread class. The new class must override the run() method to define the code that will be executed by the thread. Here's an example:

```
java
public class MyThread extends Thread {
    public void run() {
        // Code to be executed by the thread
    }
}
```

Implementing the Runnable interface: This involves creating a new class that implements the java.lang.Runnable interface. The new class must define a run() method that contains the code to be executed by the thread. This class can then be passed to the Thread constructor, which creates a new thread and starts its execution. Here's an example:

```
java
public class MyRunnable implements Runnable {
    public void run() {
        // Code to be executed by the thread
    }
}


public class MyThread implements Runnable {
    public static void main(String[] args) {
        Thread thread = new Thread(new MyRunnable());
        thread.start();
    }
}
```

8. Define Multi-threading? Construct a java program to create multiple threads in java by implementing runnable interface

Multithreading is a programming concept that allows a single process to create and manage multiple threads of execution concurrently.

```
// Define a class that implements the Runnable interface
class MyRunnable implements Runnable {
    private String threadName;

    public MyRunnable(String name) {
        this.threadName = name;
    }

    // Implement the run method to define the task for each thread
    public void run() {
        System.out.println("Thread " + threadName + " is running.");
    }
}

public class MultiThreadExample {
    public static void main(String[] args) {
        // Create multiple threads by instantiating the MyRunnable class
        Thread thread1 = new Thread(new MyRunnable("Thread 1"));
        Thread thread2 = new Thread(new MyRunnable("Thread 2"));
        Thread thread3 = new Thread(new MyRunnable("Thread 3"));

        // Start each thread
        thread1.start();
        thread2.start();
        thread3.start();
    }
}
```

10. Demonstrate i) isAlive( ) and join( ) ii)values() iii)valuesof() iv)compareTo()

isAlive() and join() Methods:

java

```
public class ThreadDemo extends Thread {
    public void run() {
        System.out.println("Thread is running.");
    }

    public static void main(String[] args) throws InterruptedException {
        ThreadDemo thread = new ThreadDemo();

        // Check if the thread is alive
        System.out.println("Is thread alive? " + thread.isAlive());

        // Start the thread
        thread.start();

        // Wait for the thread to finish using join()
        thread.join();

        // Check again if the thread is alive
        System.out.println("Is thread alive? " + thread.isAlive());
    }
}
```

compareTo() Method:

java

```
public class CompareToDemo {
    public static void main(String[] args) {
        Integer num1 = 10;
        Integer num2 = 5;

        // Compare two integers using compareTo()
        int result = num1.compareTo(num2);

        if (result > 0) {
            System.out.println(num1 + " is greater than " + num2);
        } else if (result < 0) {
            System.out.println(num1 + " is less than " + num2);
        } else {
            System.out.println(num1 + " is equal to " + num2);
        }
    }
}
```

These examples demonstrate the usage of various methods in Java:

isAlive() and join() methods for managing threads.

values() method to retrieve all enum constants.

valueOf() method to convert a string to an enum constant.

compareTo() method to compare two objects.

12. Develop a JAVA program to raise a custom exception (user defined exception) for DivisionbyZero using try , catch , throw and finally

12.

```
// Define a custom exception class for division by zero
class DivisionByZeroException extends Exception {
    public DivisionByZeroException(String message) {
        super(message);
    }
}

public class CustomExceptionDemo {
    public static void main(String[] args) {
        int dividend = 10;
        int divisor = 0;

        try {
            if (divisor == 0) {
                // Throw custom exception if divisor is zero
                throw new DivisionByZeroException("Division by zero is not
allowed.");
            } else {
                int result = dividend / divisor;
                System.out.println("Result of division: " + result);
            }
        } catch (DivisionByZeroException e) {
            System.out.println("Custom Exception Caught: " + e.getMessage());
        } finally {
            System.out.println("Finally block executed.");
        }
    }
}
```

15. Write a program to illustrare creation of threads using runnable class .(start method start each of the newly created thread .Inside the run method there is sleep() for suspend the thread for 500 milliseconds)

15.

```
class MyRunnable implements Runnable {
    public void run() {
        try {
            System.out.println("Thread is running.");
            Thread.sleep(500); // Suspend the thread for 500 milliseconds
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted.");
        }
    }

    public static void main(String[] args) {
        MyRunnable myRunnable = new MyRunnable();

        // Create multiple threads using the same instance of MyRunnable
        Thread thread1 = new Thread(myRunnable);
        Thread thread2 = new Thread(myRunnable);

        // Start each thread
        thread1.start();
        thread2.start();
    }
}
```