Ex No 4.1 Write a java program that implements a multi-threaded application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.

Aim

To write a java program that implements a multi-threaded application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.

Algorithm

1. Create a class RandomNumberGenerator - Implement Runnable interface.Generate a random integer every 1 second.If the number is even, send it to SquareCalculator thread.If the number is odd, send it to CubeCalculator thread.
2. Create a class SquareCalculator - Implement Runnable interface.Receive the even number from RandomNumberGenerator.Compute its square.Print the result.
3. Create a class CubeCalculator - Implement Runnable interface.Receive the odd number from RandomNumberGenerator.Compute its cube. Print the result.
4. Create a Main class with a main method - Create instances of RandomNumberGenerator, SquareCalculator, and CubeCalculator. Create Thread objects for each class.Start all three threads.

Program

```
import java.util.Random;

// Thread 1: Generates a random number every second
class NumberGenerator extends Thread {
   private final SharedResource resource;

   public NumberGenerator(SharedResource resource) {
      this.resource = resource;
   }

   @Override
   public void run() {
      Random random = new Random();
      while (true) {
         int num = random.nextInt(100); // Generate a random number (0 to 99)
         System.out.println("\nGenerated Number: " + num);
         resource.setNumber(num); // Notify other threads
         try {
            Thread.sleep(1000); // Wait for 1 second before generating a new number
         } catch (InterruptedException e) {
```

```java
                e.printStackTrace();
            }
        }
    }
}

// Thread 2: Computes the square of even numbers
class SquareCalculator extends Thread {
    private final SharedResource resource;

    public SquareCalculator(SharedResource resource) {
        this.resource = resource;
    }

    @Override
    public void run() {
        while (true) {
            int num = resource.getNumber();
            if (num % 2 == 0) {
                System.out.println("Square of " + num + " = " + (num * num));
            }
            try {
                Thread.sleep(1000); // Avoid unnecessary CPU usage
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

// Thread 3: Computes the cube of odd numbers
class CubeCalculator extends Thread {
    private final SharedResource resource;

    public CubeCalculator(SharedResource resource) {
        this.resource = resource;
    }

    @Override
    public void run() {
        while (true) {
            int num = resource.getNumber();
            if (num % 2 != 0) {
                System.out.println("Cube of " + num + " = " + (num * num * num));
            }
            try {
                Thread.sleep(1000); // Avoid unnecessary CPU usage
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

// Shared resource to pass data between threads
class SharedResource {
    private int number;

    public synchronized void setNumber(int number) {
        this.number = number;
    }

    public synchronized int getNumber() {
        return number;
    }
}

// Main class
public class MultiThreadedApp {
    public static void main(String[] args) {
        SharedResource resource = new SharedResource();

        NumberGenerator generator = new NumberGenerator(resource);
        SquareCalculator squareCalculator = new SquareCalculator(resource);
        CubeCalculator cubeCalculator = new CubeCalculator(resource);

        // Start the threads
        generator.start();
        squareCalculator.start();
        cubeCalculator.start();
    }
}
```

Output

```
Generated Number: 42
Square of 42 = 1764

Generated Number: 15
Cube of 15 = 3375

Generated Number: 88
```

```
Square of 88 = 7744

Generated Number: 27
Cube of 27 = 19683

Generated Number: 56
Square of 56 = 3136

Generated Number: 91
Cube of 91 = 753571
```

Result

Thus the java program that implements a multi-threaded application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number that has been completed successfully and the result is verified.

Ex No 4.2 Write a program to implement the concept of importing classes from user defined package and creating packages

Aim

To write a program to implement the concept of importing classes from user defined package and creating packages.

Algorithm

1. Create Classes inside the Package: Define one or more classes in the package. Implement required functionality in these classes.
2. Create a Package: Define a package using the package keyword. Save the file inside a folder with the package name.
3. Compile the Package: Use javac -d . ClassName.java to compile and store the package properly.
4. Import the Package in Another Java File: Use import package_name.ClassName; to access the class from the package.
5. Use the Imported Class in the Main Program: Create an instance of the imported class. Call its methods and execute the program.

Program

Java program that demonstrates:

1. **Creating a user-defined package** (mypackage).
2. **Defining a class (Calculator) inside the package** with arithmetic operations.
3. **Importing and using the package in another class (MainApp).**

Steps to Run the Program

1. **Create a folder** where you want to store your Java files.
2. **Inside that folder, create a subfolder** named mypackage (this will be the package).
3. **Save the first file (Calculator.java) inside the mypackage folder.**
4. **Save the second file (MainApp.java) outside the mypackage folder.**
5. **Compile and run MainApp.java.**

Create the Package (mypackage) and Define a Class

Save the following code as **Calculator.java** inside the mypackage folder.

```
// Declare the package
package mypackage;

// Define a class with arithmetic operations
```

```
public class Calculator {

    public int add(int a, int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }

    public int multiply(int a, int b) {
        return a * b;
    }

    public double divide(int a, int b) throws ArithmeticException {
        if (b == 0) {
            throw new ArithmeticException("Cannot divide by zero");
        }
        return (double) a / b;
    }
}
```

Import and Use the Package

Save the following code as MainApp.java outside the mypackage folder.

```
// Import the user-defined package
import mypackage.Calculator;

import java.util.Scanner;

public class MainApp {
    public static void main(String[] args) {
        // Create an object of Calculator class
        Calculator calc = new Calculator();
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter first number: ");
        int num1 = scanner.nextInt();

        System.out.print("Enter second number: ");
        int num2 = scanner.nextInt();

        System.out.println("Addition: " + calc.add(num1, num2));
        System.out.println("Subtraction: " + calc.subtract(num1, num2));
        System.out.println("Multiplication: " + calc.multiply(num1, num2));
        try {
            System.out.println("Division: " + calc.divide(num1, num2));
        } catch (ArithmeticException e) {
            System.out.println("Error: " + e.getMessage());
        }

        scanner.close();
    }
}
```

Output
Compile and Run the Program

Compiling the Package (mypackage)

Navigate to the folder where mypackage is located and compile the Calculator.java file:

```
javac -d . mypackage/Calculator.java
```

This -d . option ensures that mypackage is created as a package.

Compiling and Running MainApp.java

Compile MainApp.java:

```
javac MainApp.java
```

Run the program:

```
java MainApp
```

```
Enter first number: 10
Enter second number: 5
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
```

Result
Thus the program to implement the concept of importing classes from user defined package and creating packages has been completed successfully, and the output is verified.

Ex No 4.3 Write a java program to find the maximum value from the given type of elements using a generic function

Aim
To write a java program to find the maximum value from the given type of elements using a generic function

Algorithm

1. Define a Generic Method: Create a method that takes an array of any comparable type (e.g., Integer, Double, String). Use a type parameter <T extends Comparable<T>> to ensure elements can be compared.
2. Initialize Maximum Value: Assume the first element is the maximum.
3. Iterate Through the Array: Compare each element with the current maximum using compareTo(). If an element is greater, update the maximum.
4. Return the Maximum Value: Once all elements are checked, return the maximum.
5. Test the Generic Method in main Method: Call the method with different data types (Integer, Double, String).

Program
```
// Generic class with a method to find the maximum element
class MaxFinder {

    // Generic method to find maximum in an array
    public static <T extends Comparable<T>> T findMax(T[] array) {
        if (array == null || array.length == 0) {
            throw new IllegalArgumentException("Array is empty or null");
        }

        T max = array[0]; // Assume first element is max
        for (T element : array) {
            if (element.compareTo(max) > 0) {
                max = element; // Update max if current element is greater
            }
        }
        return max;
    }
}

public class GenericMaxFinder {
    public static void main(String[] args) {

        // Integer array
        Integer[] intArray = {10, 25, 5, 88, 42};
        System.out.println("Max Integer: " + MaxFinder.findMax(intArray));

        // Double array
        Double[] doubleArray = {12.5, 7.8, 20.9, 2.3};
        System.out.println("Max Double: " + MaxFinder.findMax(doubleArray));

        // String array (compares lexicographically)
        String[] stringArray = {"Apple", "Orange", "Banana", "Peach"};
        System.out.println("Max String: " + MaxFinder.findMax(stringArray));
    }
}
```

Output

```
Max Integer: 88
Max Double: 20.9
Max String: Peach
```

Result

Thus the java program to find the maximum value from the given type of elements using a generic function has been completed successfully and the output is verified.