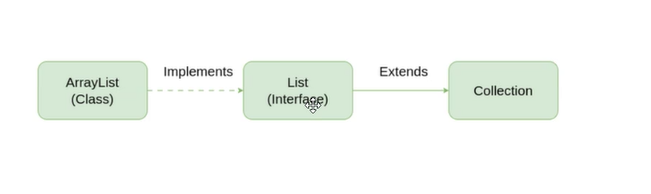
class ----->  class ======> extends  ====> no multiple inheritance  
  
interface ----> class =====> Implements  
  
interface ---> interface ====> extends



Exceptions:   --> unexpected situations

  2 categoried  
    - Checked  ---> compile time  
    - Unchecked  ---> Runtime time..

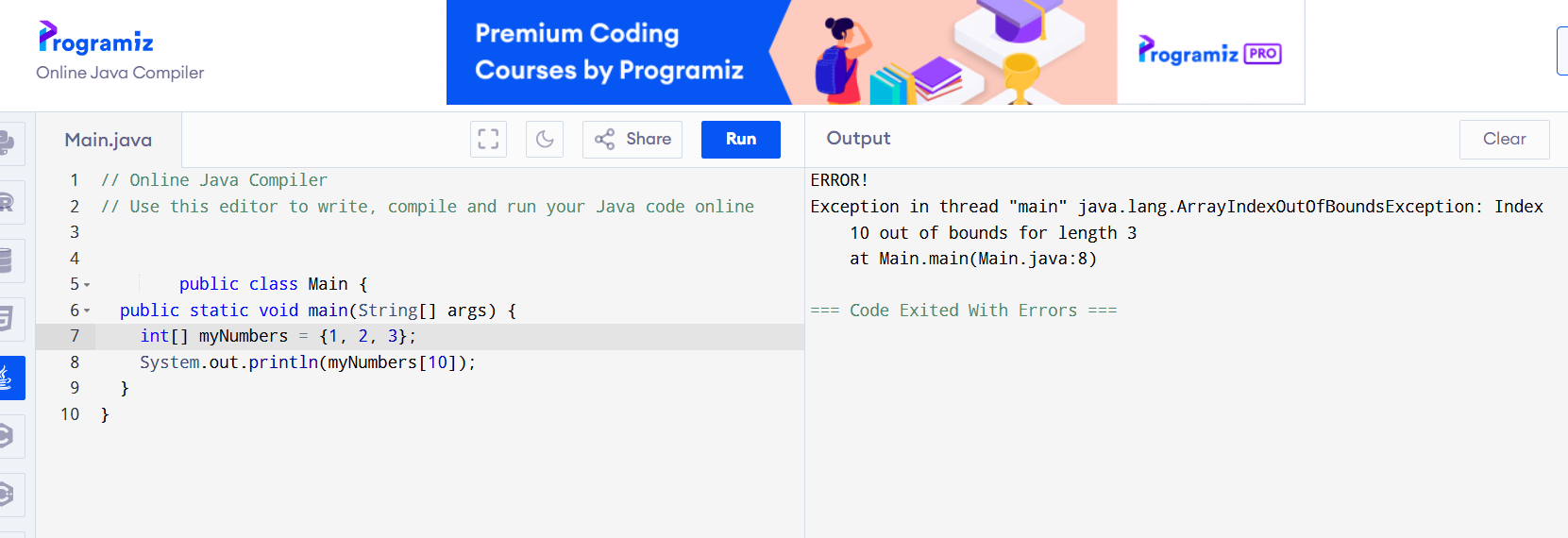
  pre defined exception classes  
    FileNotFound  
    ArrayIndexoutofBounds exception  
    NullPointer Exception  
    Arthematic Exception ...  
  
  user defined Exceptions  
  
try catch blocks  
  try{

  }catch(Exception ex){

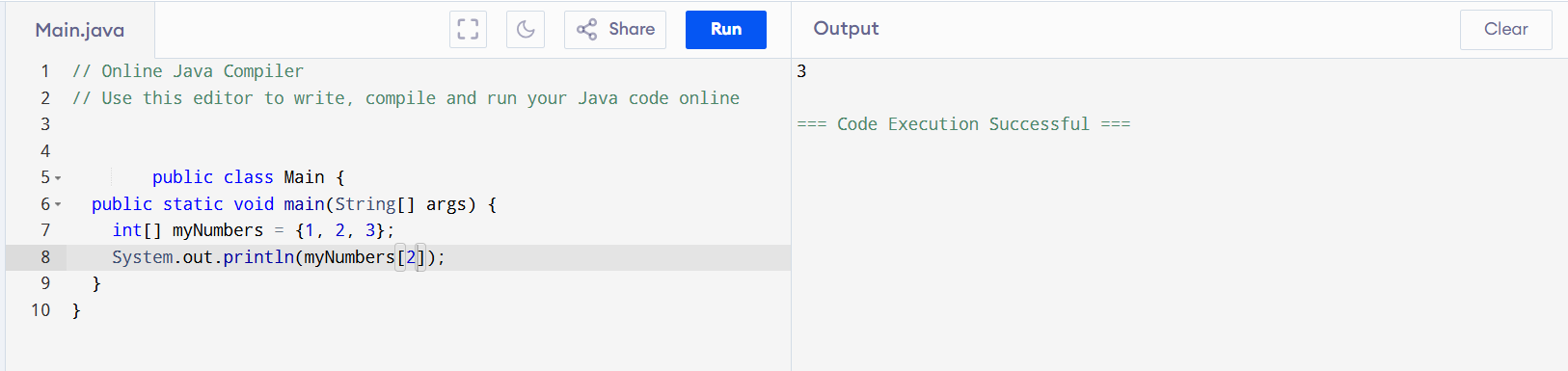
  }catch(ArthematicException ex1{  
  
  }  
  finally{

  sout(" ");  
  }

TASK1:



public class Main {  
  public static void main(String[] args) {  
    int[] myNumbers = {1, 2, 3};  
    System.out.println(myNumbers[10]);  
  }  
}



public class Main {

public static void main(String[] args) {

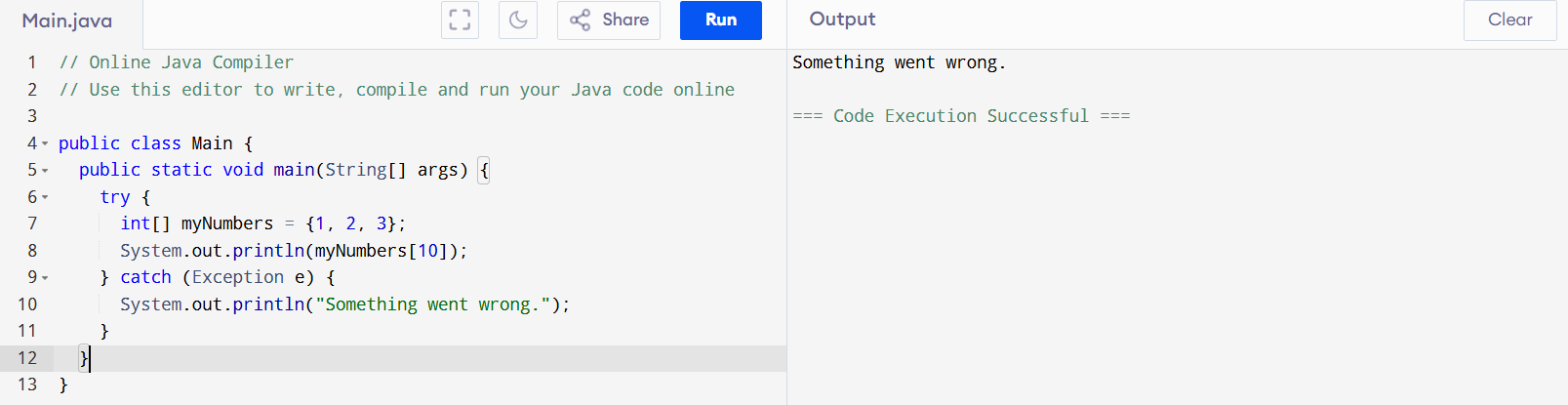
int[] myNumbers = {1, 2, 3};

System.out.println(myNumbers[2]);

}

}

TASK3:



public class Main {

public static void main(String[] args) {

try {

int[] myNumbers = {1, 2, 3};

System.out.println(myNumbers[10]);

} catch (Exception e) {

System.out.println("Something went wrong.");

}

}

}

TASK4:



public class Main {

public static void main(String[] args) {

try {

int[] myNumbers = {1, 2, 3};

System.out.println(myNumbers[10]);

} catch (Exception e) {

System.out.println("Something went wrong.");

} finally {

System.out.println("I'm from finally block.");

}

}

}



public class Main {

public static void main(String[] args) {

try {

int[] myNumbers = {1, 2, 3};

System.out.println(myNumbers[2]);

} catch (Exception e) {

System.out.println("Something went wrong.");

} finally {

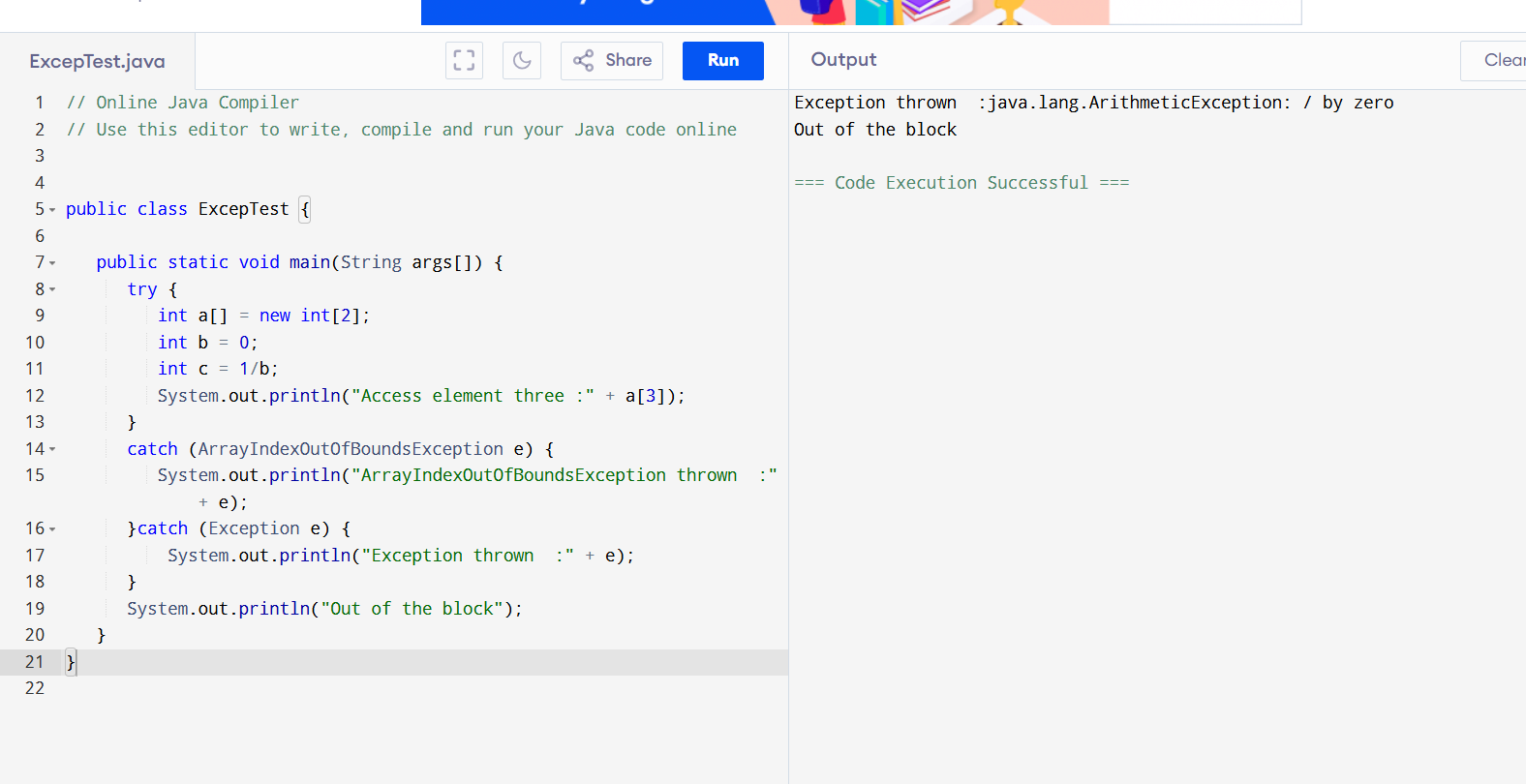
System.out.println("I'm from finally block.");

}

}

}

Task:5



public class ExcepTest {

public static void main(String args[]) {

try {

int a[] = new int[2];

int b = 0;

int c = 1/b;

System.out.println("Access element three :" + a[3]);

}

catch (ArrayIndexOutOfBoundsException e) {

System.out.println("ArrayIndexOutOfBoundsException thrown :" + e);

}catch (Exception e) {

System.out.println("Exception thrown :" + e);

}

System.out.println("Out of the block");

}

}



public class ExcepTest {

public static void main(String args[]) {

try {

int a[] = new int[2];

int b = 0;

int c = 1/b;

System.out.println("Access element three :" + a[3]);

}

catch (ArrayIndexOutOfBoundsException e) {

System.out.println("ArrayIndexOutOfBoundsException thrown :" + e);

}catch (Exception e) {

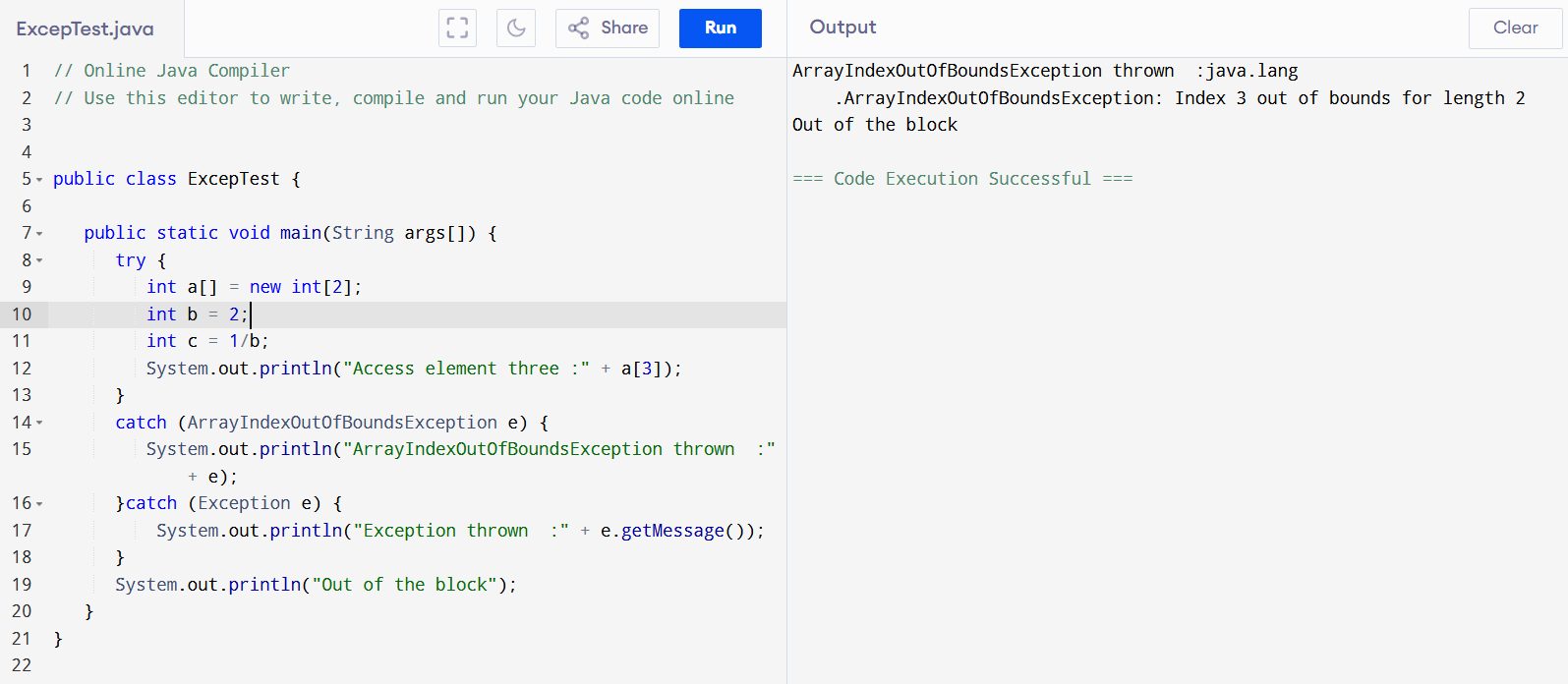
System.out.println("Exception thrown :" + e.getMessage());

}

System.out.println("Out of the block");

}

}



public class ExcepTest {

public static void main(String args[]) {

try {

int a[] = new int[2];

int b = 2;

int c = 1/b;

System.out.println("Access element three :" + a[3]);

}

catch (ArrayIndexOutOfBoundsException e) {

System.out.println("ArrayIndexOutOfBoundsException thrown :" + e);

}catch (Exception e) {

System.out.println("Exception thrown :" + e.getMessage());

}

System.out.println("Out of the block");

}

}

TASK:6

public class ExcepTest {

public static void main(String args[]) {

try {

int a[] = new int[2];

System.out.println("Access element three :" + a[0]);

try {

int b = 1;

int c = 1/b;

}catch(Exception e) {

System.out.println("Exception thrown: " + e);

}

System.out.println("Access element three :" + a[0]);

}

catch (ArrayIndexOutOfBoundsException e) {

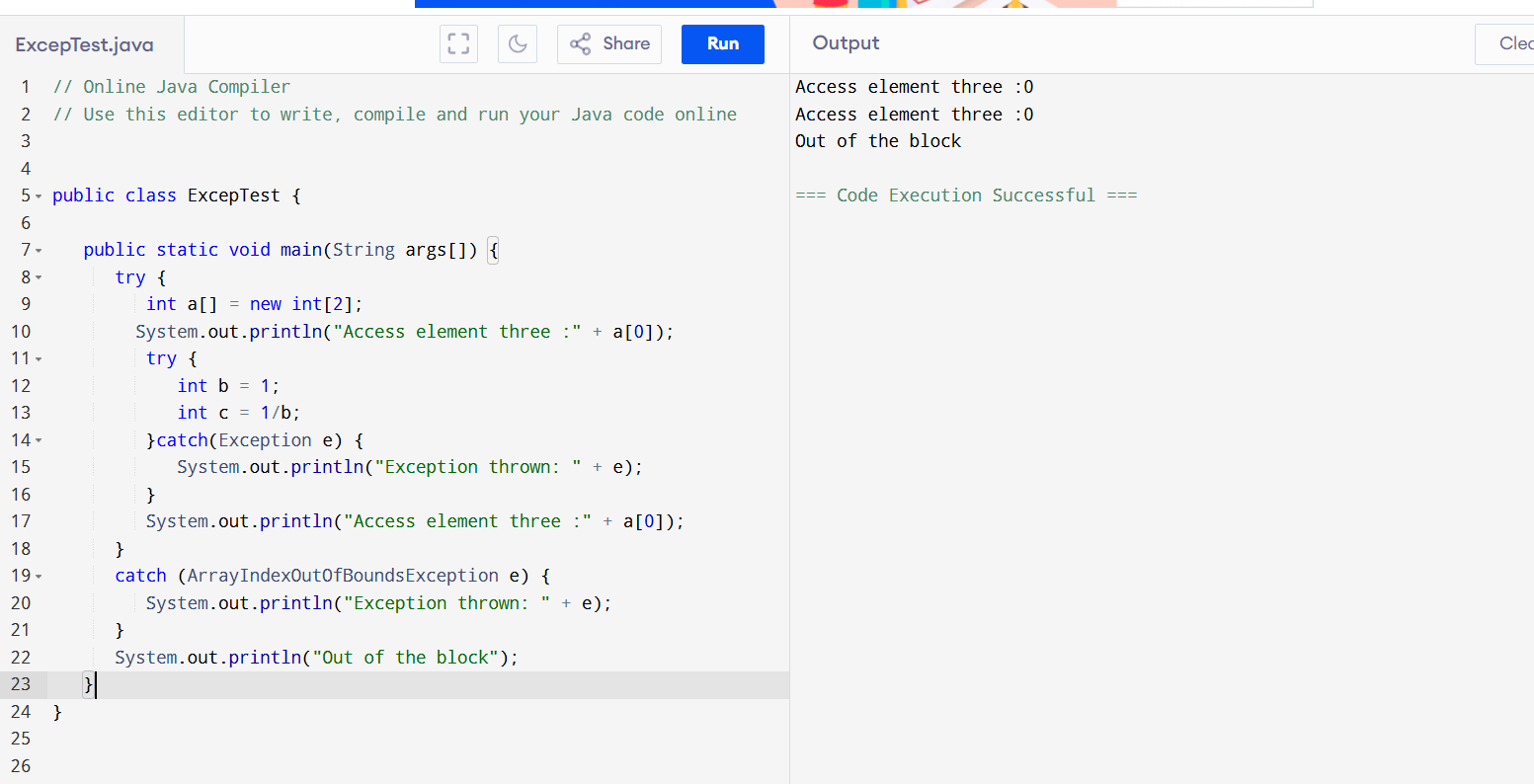
System.out.println("Exception thrown: " + e);

}

System.out.println("Out of the block");

}

}



public class ExcepTest {

public static void main(String args[]) {

try {

int a[] = new int[2];

System.out.println("Access element three :" + a[2]); // try with a[0] or a[1] ===> and check if control goes to inner try block..

try {

int b = 0;

int c = 1/b;

}catch(Exception e) {

System.out.println("Exception thrown: " + e);

}

System.out.println("Access element three :" + a[3]);

}

catch (ArrayIndexOutOfBoundsException e) {

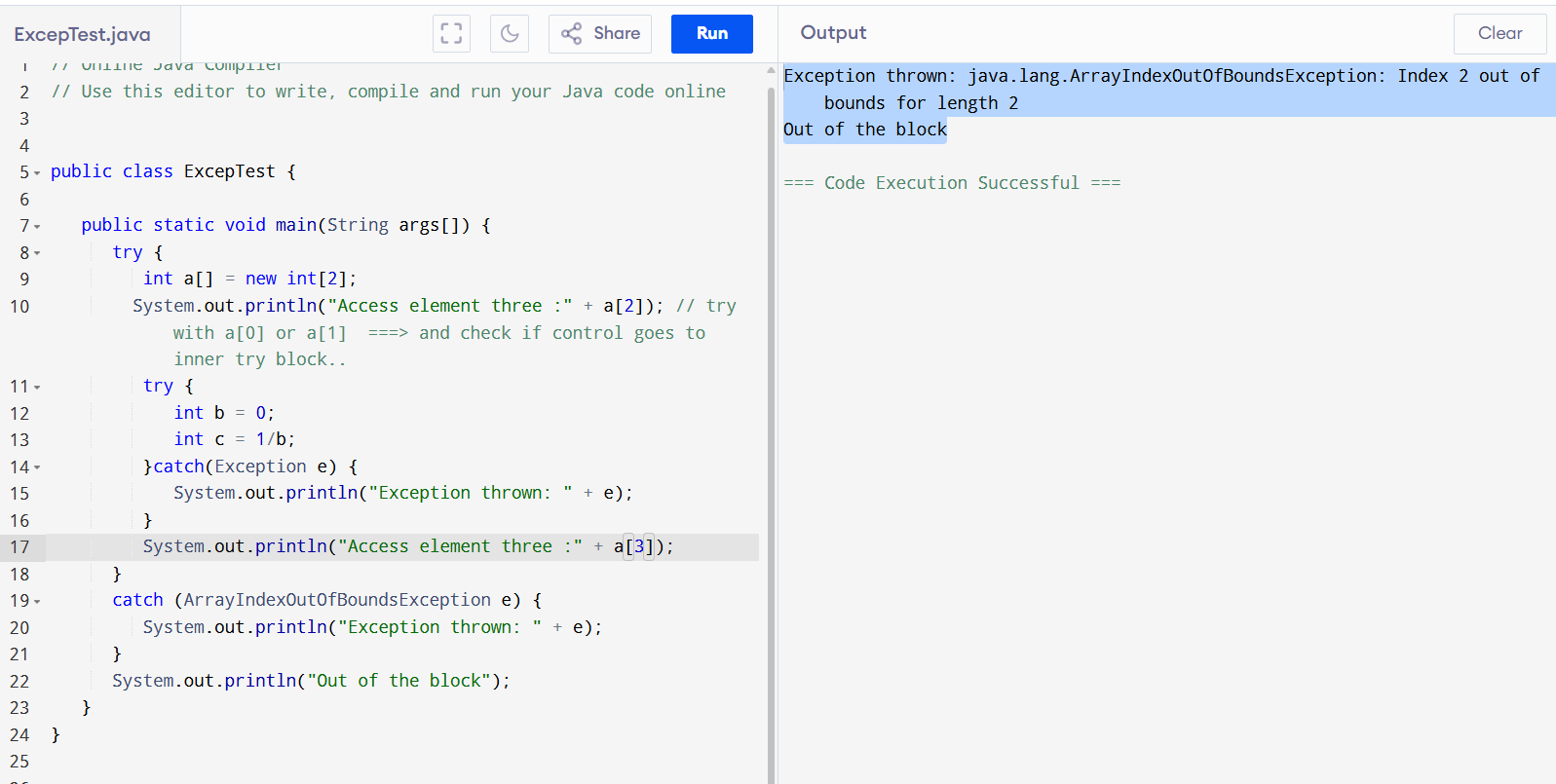
System.out.println("Exception thrown: " + e);

}

System.out.println("Out of the block");

}

}



Exception with out catch:

public class Example {

public static void main(String[] args) {

int result = 10 / 0; // This will throw ArithmeticException

System.out.println("Result: " + result);

System.out.println("Error: Cannot divide by zero.");

}

}



With Catch:

public class Example {

public static void main(String[] args) {

int result = 10 / 0; // This will throw ArithmeticException

System.out.println("Result: " + result);

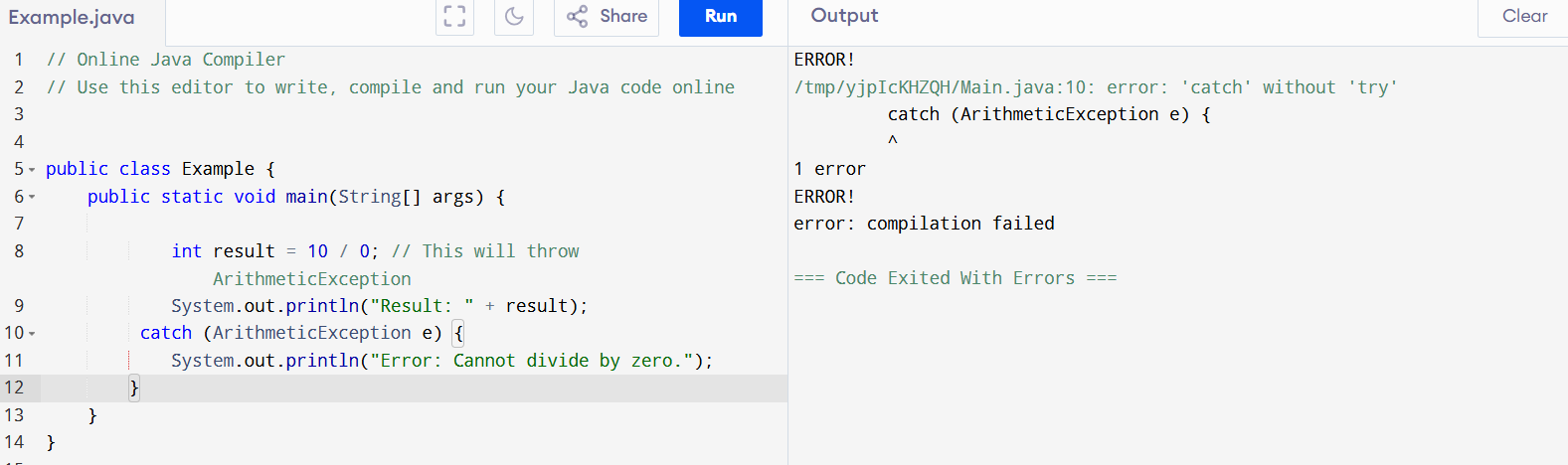
catch (ArithmeticException e) {

System.out.println("Error: Cannot divide by zero.");

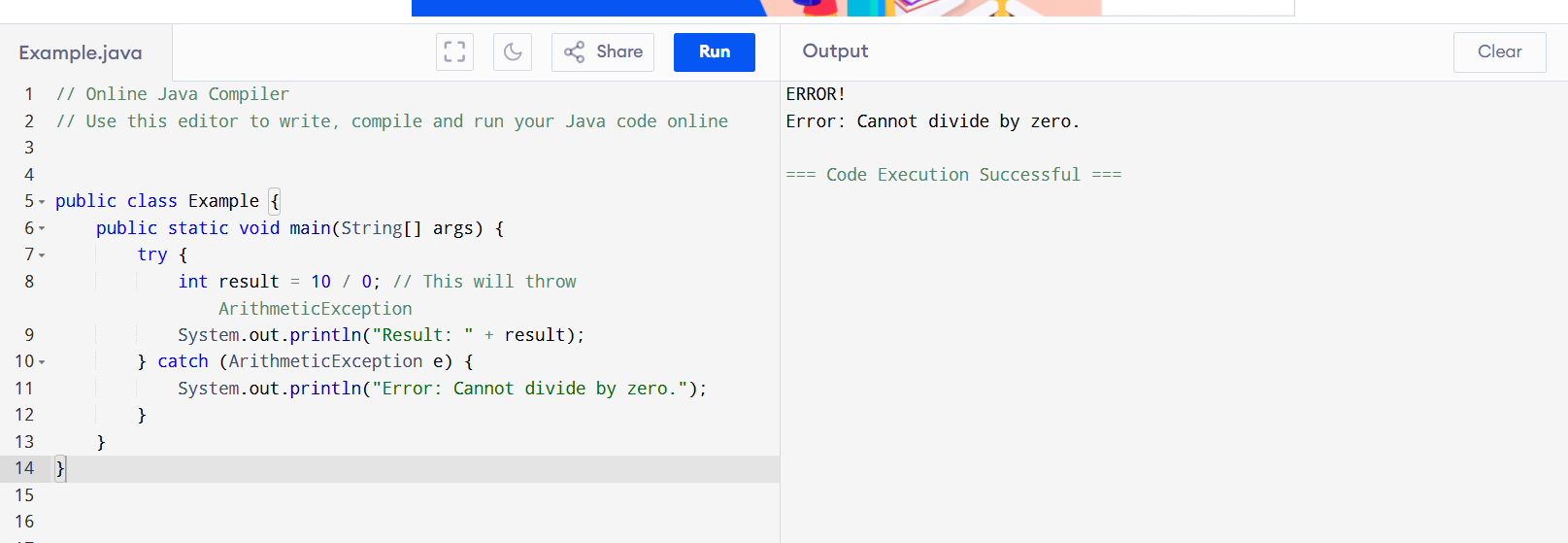
}

}

}



With Try:



public class Example {

public static void main(String[] args) {

try {

int result = 10 / 0; // This will throw ArithmeticException

System.out.println("Result: " + result);

} catch (ArithmeticException e) {

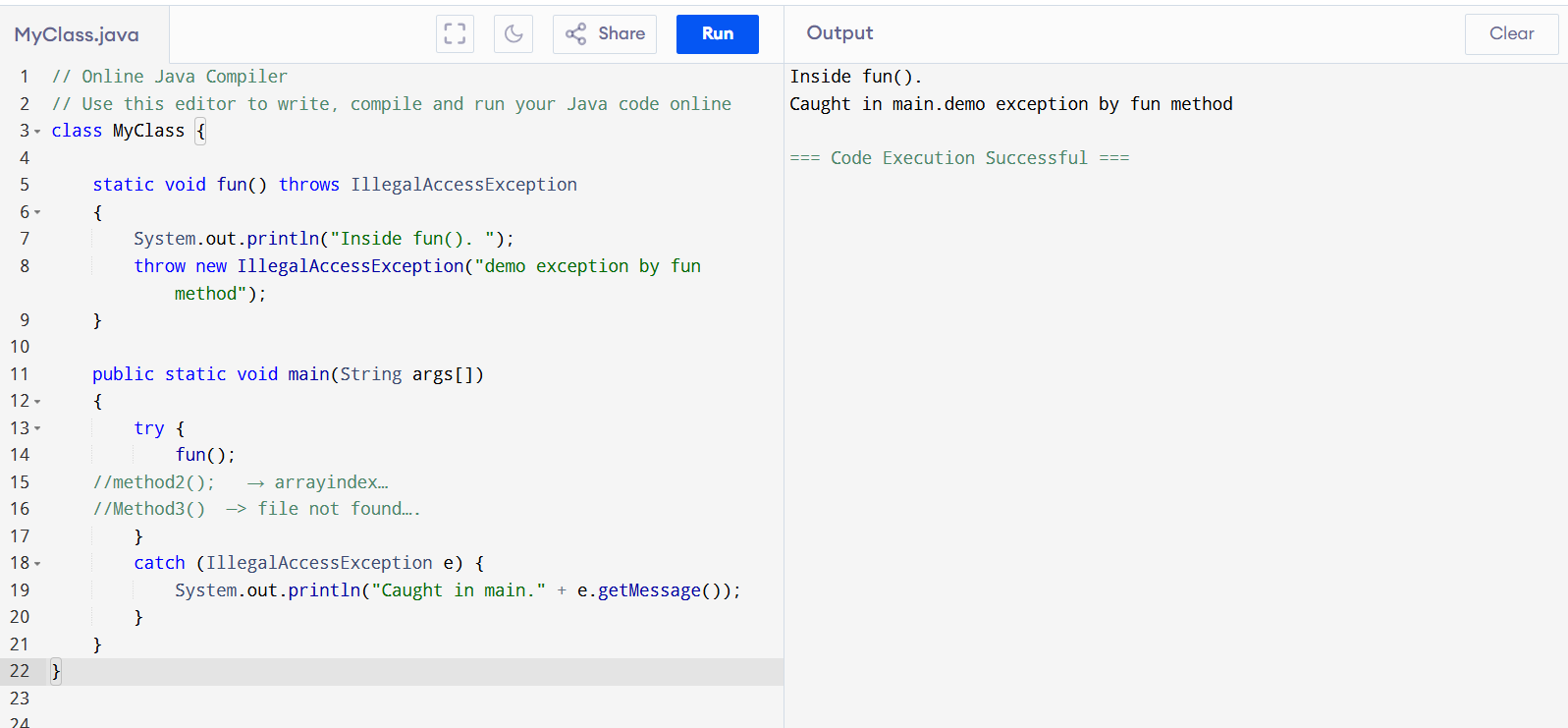
System.out.println("Error: Cannot divide by zero.");

}

}

}

TASK:7



class MyClass {

static void fun() throws IllegalAccessException

{

System.out.println("Inside fun(). ");

throw new IllegalAccessException("demo exception by fun method");

}

public static void main(String args[])

{

try {

fun();

//method2(); → arrayindex…

//Method3() —> file not found….

}

catch (IllegalAccessException e) {

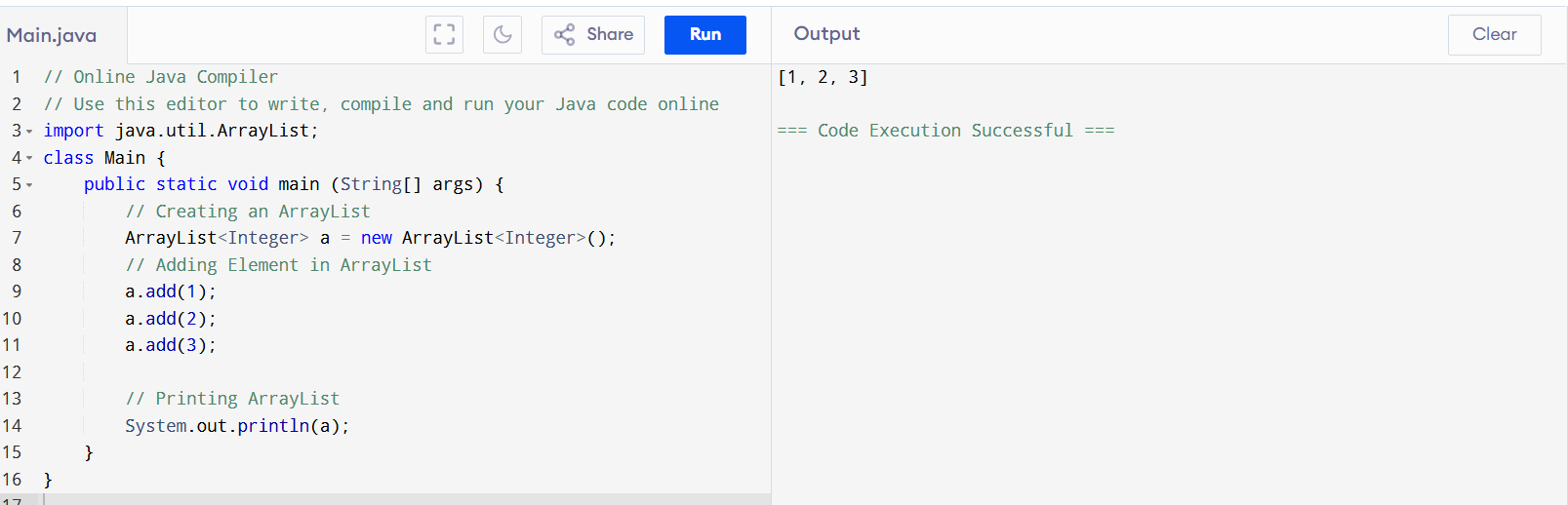
System.out.println("Caught in main." + e.getMessage());

}

}

}

TASK:9



import java.util.ArrayList;

class Main {

public static void main (String[] args) {

// Creating an ArrayList

ArrayList<Integer> a = new ArrayList<Integer>();

// Adding Element in ArrayList

a.add(1);

a.add(2);

a.add(3);

// Printing ArrayList

System.out.println(a);

}

}

TASK 10:

import java.util.\*;

class Main {

public static void main(String args[]){

ArrayList<String> al = new ArrayList<>();

al.add("Prasunamba");

al.add("Meher");

System.out.println("Original List : "+al);

al.add(1, "Hello");

System.out.println("After Adding element at index 1 : "+ al);

al.remove(0);

System.out.println("Element removed from index 0 : "+ al);

al.remove("Prasunamba");

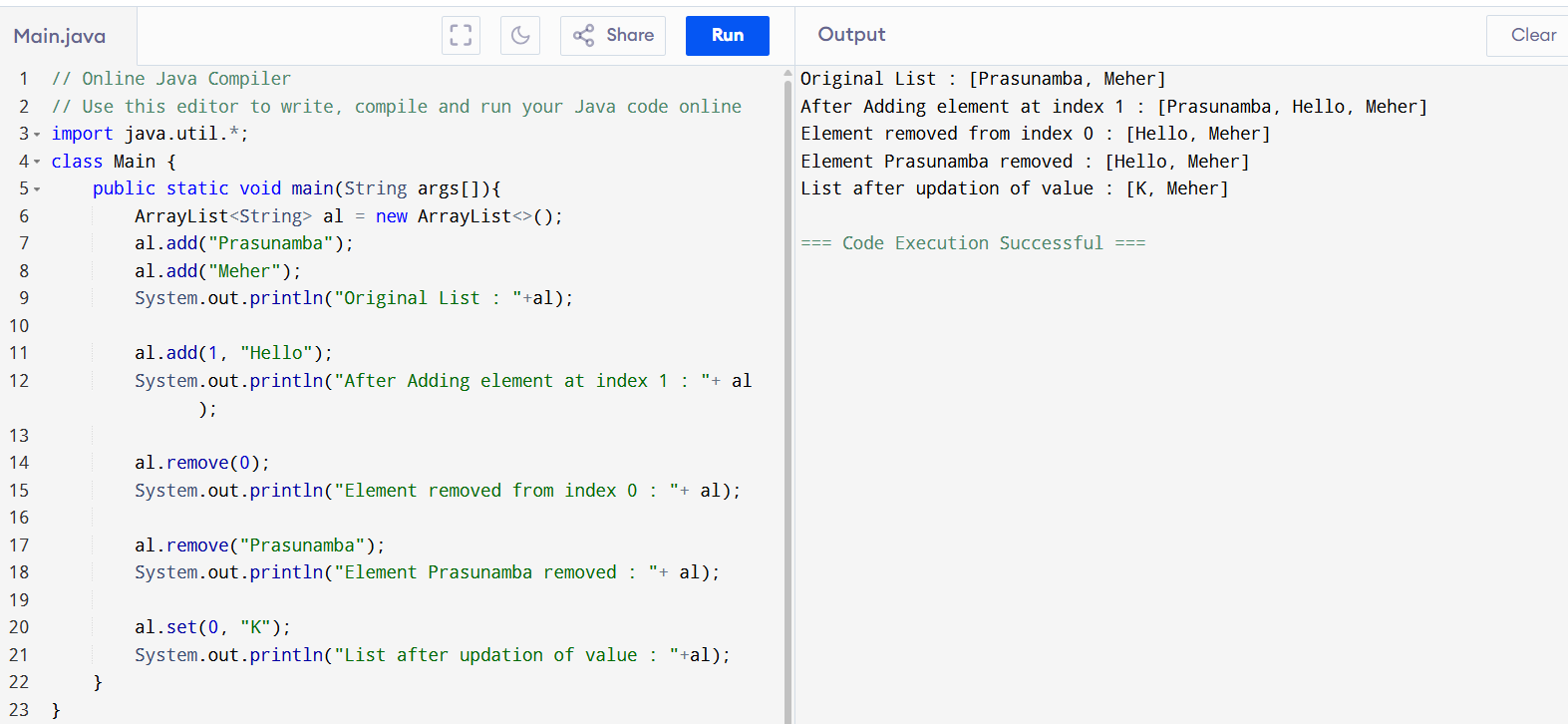
System.out.println("Element Prasunamba removed : "+ al);

al.set(0, "K");

System.out.println("List after updation of value : "+al);

}

}



TASK11:

class OuterClass {

  int x = 10;

  class InnerClass {

    int y = 5;

  }

}

public class Main {

  public static void main(String[] args) {

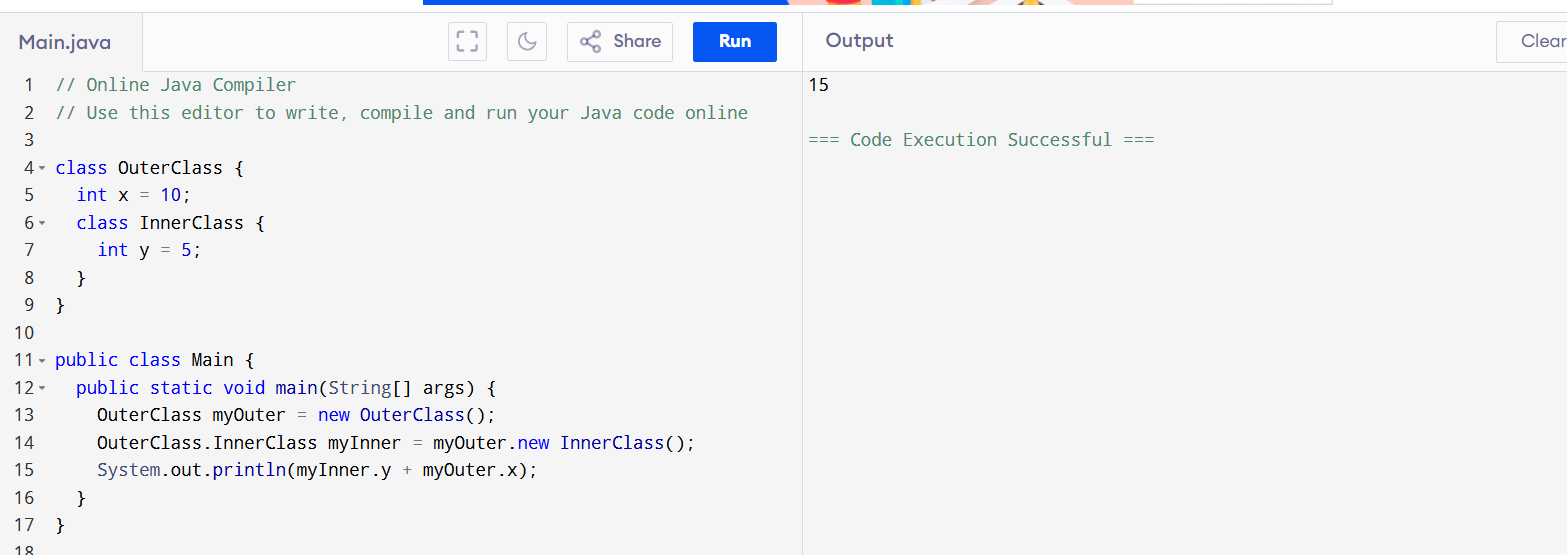
    OuterClass myOuter = new OuterClass();

    OuterClass.InnerClass myInner = myOuter.new InnerClass();

    System.out.println(myInner.y + myOuter.x);

  }

}



With Private class:

class OuterClass {

int x = 10;

private class InnerClass {

int y = 5;

}

}

public class Main {

public static void main(String[] args) {

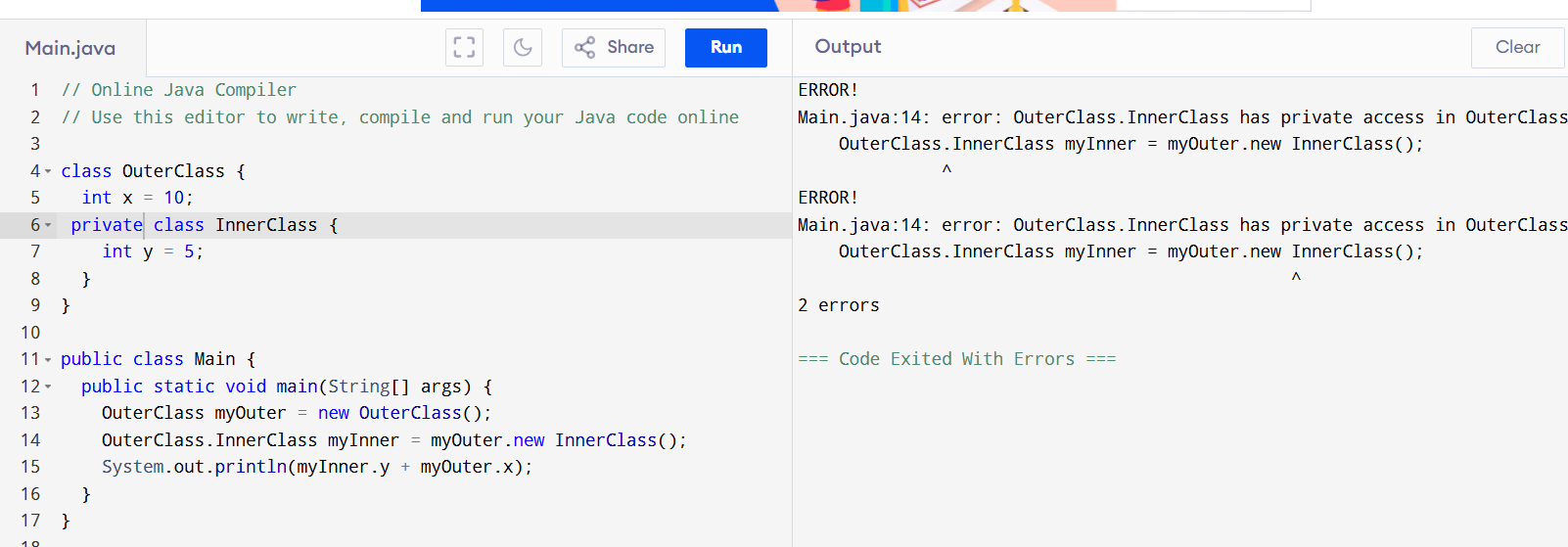
OuterClass myOuter = new OuterClass();

OuterClass.InnerClass myInner = myOuter.new InnerClass();

System.out.println(myInner.y + myOuter.x);

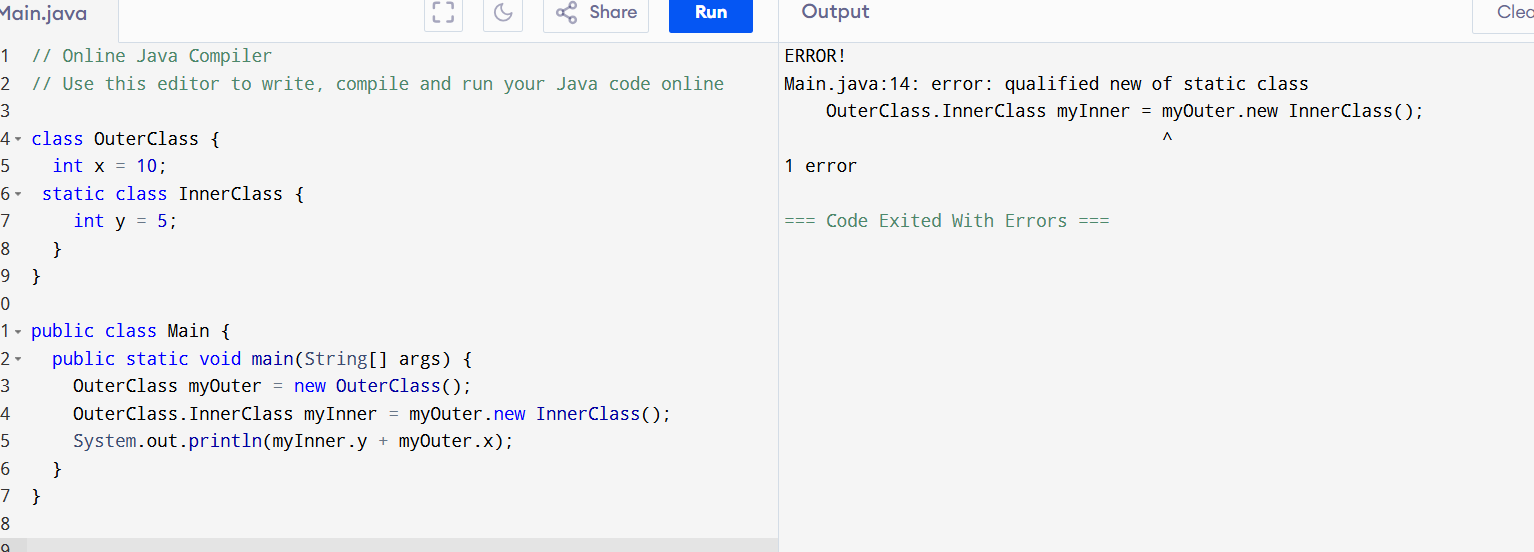
}

}



Task 13:

Use the above code Task 011 and make the inner class static



To fix task13 error:

class OuterClass {

int x = 10;

static class InnerClass {

int y = 5;

}

}

public class Main {

public static void main(String[] args) {

OuterClass myOuter = new OuterClass();

// OuterClass.InnerClass myInner = myOuter.new InnerClass();

OuterClass.InnerClass myInner = new OuterClass.InnerClass();

System.out.println(myInner.y + myOuter.x);

}

}



Task: 14

class OuterClass {

int x = 50;

class InnerClass {

public int innerMethod() {

return x;

}

}

}

public class DriverClass {

public static void main(String[] args) {

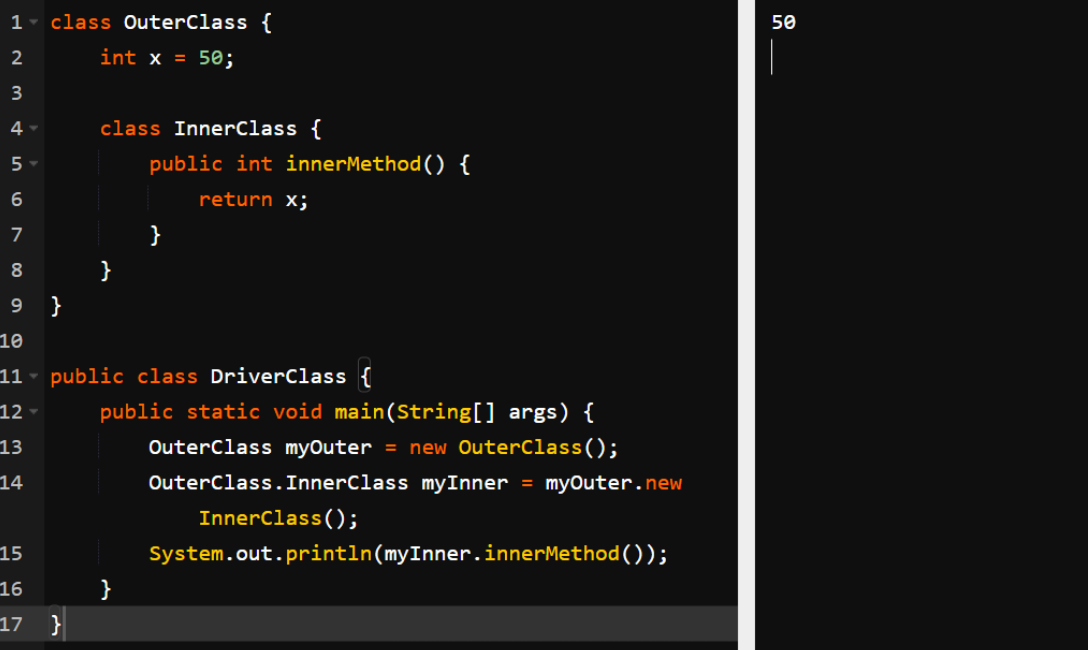
OuterClass myOuter = new OuterClass();

OuterClass.InnerClass myInner = myOuter.new InnerClass();

System.out.println(myInner.innerMethod());

}

}



TASK: 15

class OuterClass {

  int x = 10;

  static class InnerClass {

    static int y = 5;

  }

}

public class Main {

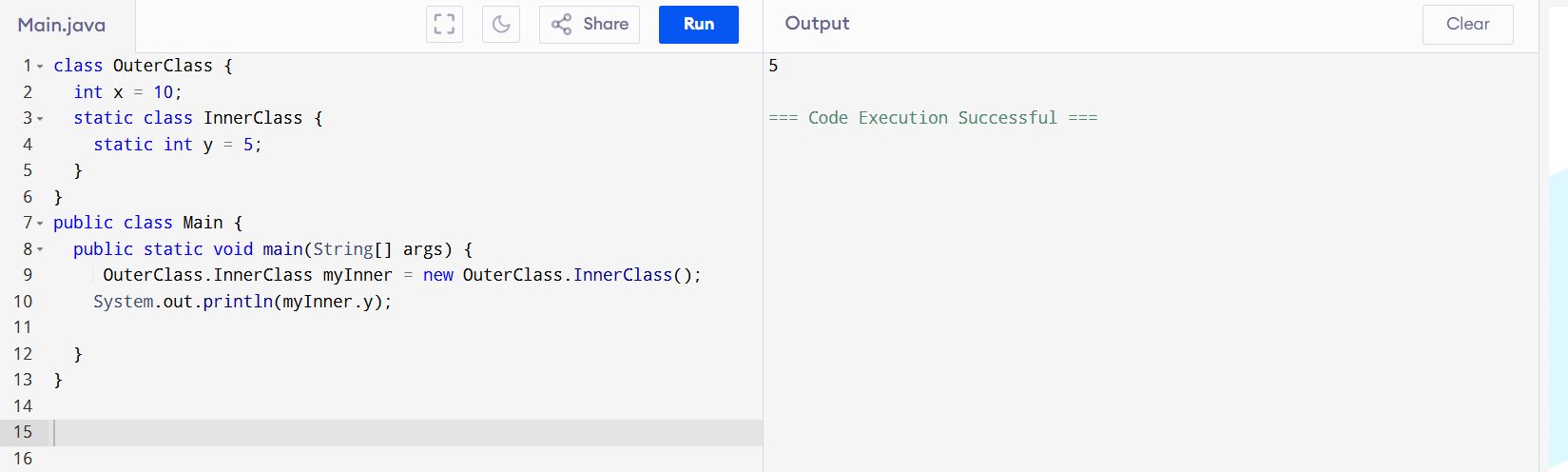
  public static void main(String[] args) {

     OuterClass.InnerClass myInner = new OuterClass.InnerClass();

    System.out.println(myInner.y);

  }

}



Lamda expressions- used to simple coding-reduces few lines of code.

TASK 17:

Features of java8:

Lambda Expressions:

Functional Interfaces: Single-method interfaces.

Introduced and Improved APIs:

Stream API: Efficient Data Manipulation.

Date/Time API: Robust Date and Time Handling.

Collection API Improvements: Enhanced Methods for Collections (e.g., removeIf, replaceAll).

Concurrency API Improvements: New classes for parallel processing (e.g., CompletableFuture).

Optional Class: Handle null values safely.

forEach() Method in Iterable Interface: Executes an action for each element in a Collection.

Default Methods Features of java8:

Lambda Expressions:

Functional Interfaces

Introduced and Improved APIs

Stream API

Date/Time API

Collection API Improvement

Concurrency API Improvements

Optional Class

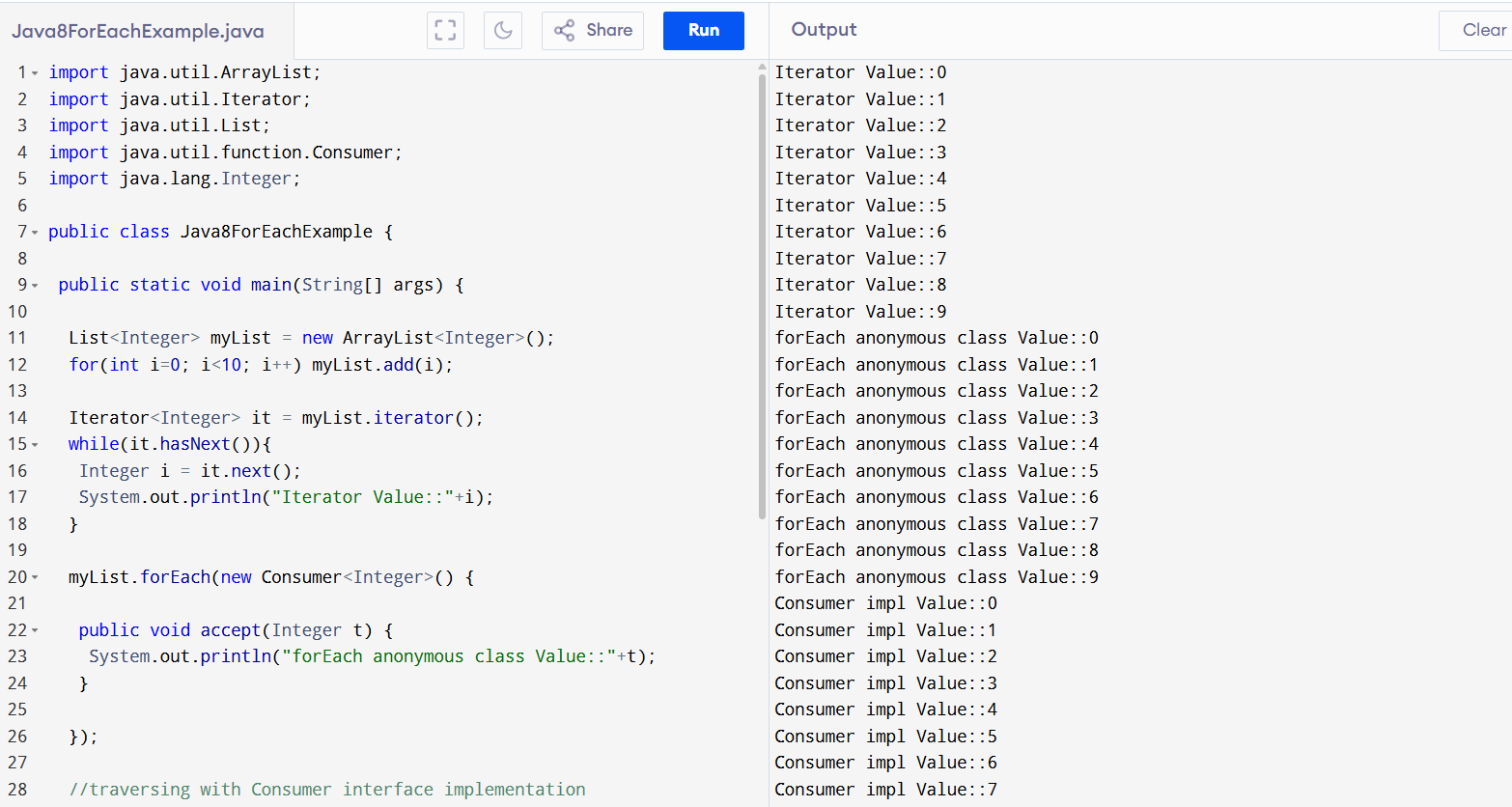
forEach() Method in Iterable Interface

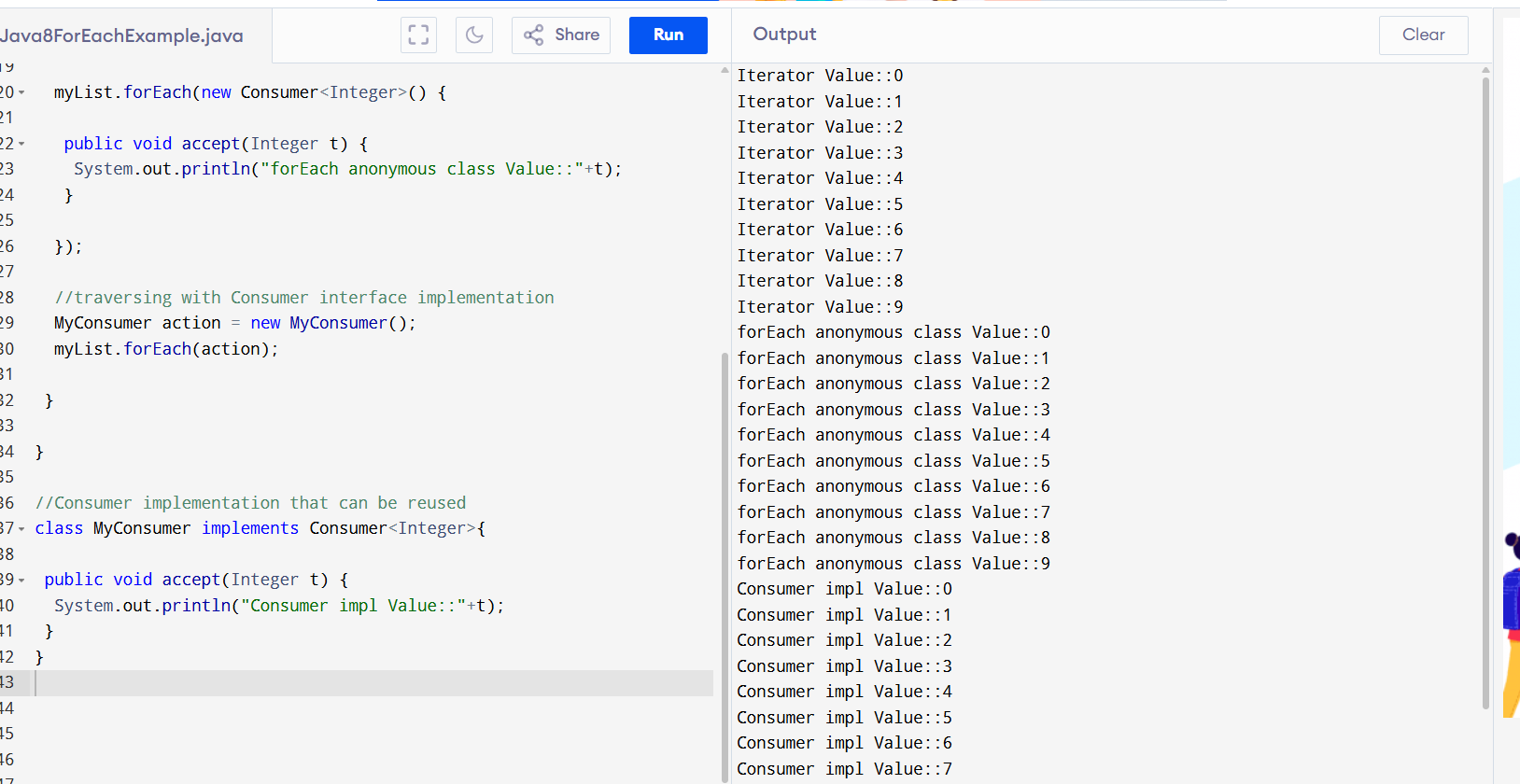
Evolve interfaces without breaking compatibility.

Static Methods

Method References

TASK18:





import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.function.Consumer;

import java.lang.Integer;

public class Java8ForEachExample {

 public static void main(String[] args) {

  List<Integer> myList = new ArrayList<Integer>();

  for(int i=0; i<10; i++) myList.add(i);

  Iterator<Integer> it = myList.iterator();

  while(it.hasNext()){

   Integer i = it.next();

   System.out.println("Iterator Value::"+i);

  }

  myList.forEach(new Consumer<Integer>() {

   public void accept(Integer t) {

    System.out.println("forEach anonymous class Value::"+t);

   }

  });

  //traversing with Consumer interface implementation

  MyConsumer action = new MyConsumer();

  myList.forEach(action);

 }

}

//Consumer implementation that can be reused

class MyConsumer implements Consumer<Integer>{

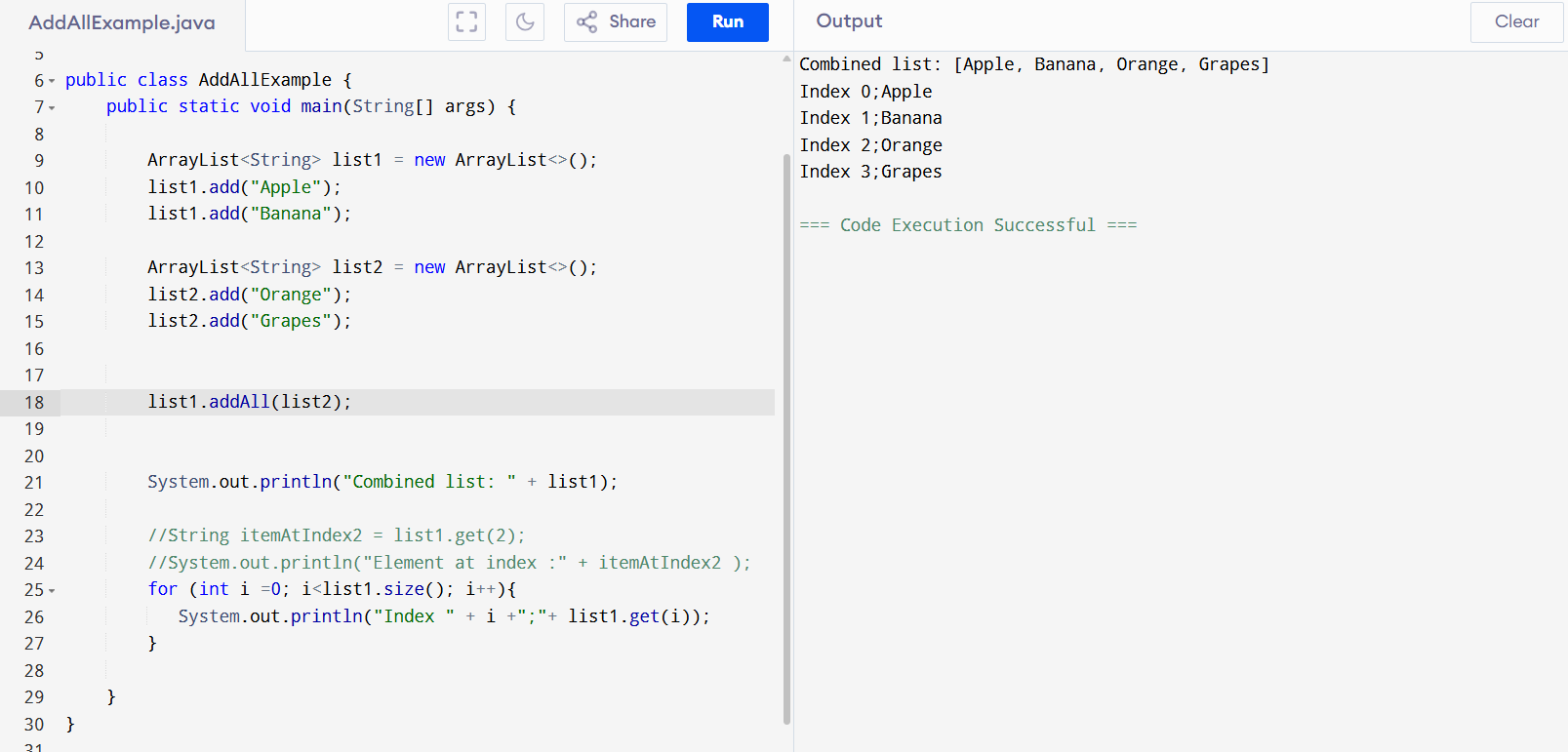
 public void accept(Integer t) {

  System.out.println("Consumer impl Value::"+t);

 }

}

Code with addAll:



public class AddAllExample {

public static void main(String[] args) {

ArrayList<String> list1 = new ArrayList<>();

list1.add("Apple");

list1.add("Banana");

ArrayList<String> list2 = new ArrayList<>();

list2.add("Orange");

list2.add("Grapes");

list1.addAll(list2);

System.out.println("Combined list: " + list1);

//String itemAtIndex2 = list1.get(2);

//System.out.println("Element at index :" + itemAtIndex2 );

for (int i =0; i<list1.size(); i++){

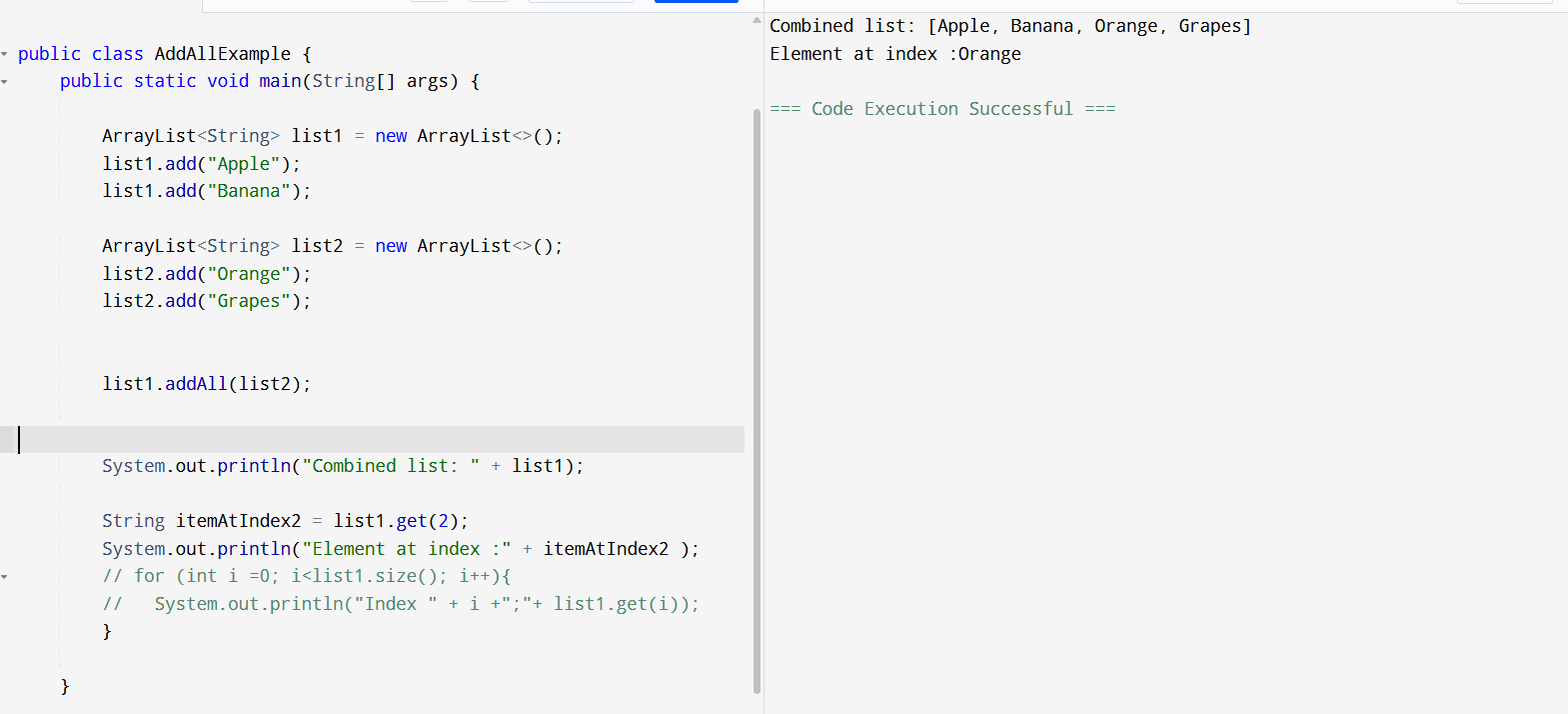
System.out.println("Index " + i +";"+ list1.get(i));

}

}

}

With Index:



public class AddAllExample {

public static void main(String[] args) {

ArrayList<String> list1 = new ArrayList<>();

list1.add("Apple");

list1.add("Banana");

ArrayList<String> list2 = new ArrayList<>();

list2.add("Orange");

list2.add("Grapes");

list1.addAll(list2);

System.out.println("Combined list: " + list1);

String itemAtIndex2 = list1.get(2);

System.out.println("Element at index :" + itemAtIndex2 );

// for (int i =0; i<list1.size(); i++){

// System.out.println("Index " + i +";"+ list1.get(i));

}

}

‑