Task 1:

Code:

public class Main {

public static void main(String[] args) {

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

System.out.println(myNumbers[1]);

}

}

Output:

2

=== Code Execution Successful ===

Task 2

Code:

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.");

}

}

}

Output:

Something went wrong.

=== Code Execution Successful ===

Task 3

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.");

}

}

}

Output

Something went wrong.

I'm from finally block.

=== Code Execution Successful ===

Task 4:

from the above code from task 3 .. try accessing the 2 or the 3rd index and check the out..

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.");

}

}

}

Output:

3

I'm from finally block.

Task 3\_1:

public class Main {

  public static void main(String[] args) {

   int[] myNumbers = {1, 2, 3};

try {

   System.out.println(myNumbers[10]);

} catch (Exception e) {

   System.out.println("Something went wrong."+ myNumbers[1]);

} finally {

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

}

  }

}

Output:

Something went wrong.2

I'm from finally block.

=== Code Execution Successful ===

Task 5:

Try with Multiple catch blocks  …. Execute the below code snippet n display the out .. along with reason..

public class task005exception {  
  
 public static void main(String args[]) {  
 try {  
 int a[] = new int[2];  
 int b = 0; //⇒ change 0 to 10 and check..  
 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");  
 }  
}

output

Exception thrown :/ by zero

Out of the block

Process finished with exit code 0

Task 6:

Nested try blocks

Output:

Exception thrown: java.lang.ArrayIndexOutOfBoundsException: Index 2 out of bounds for length 2

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 {

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");

}

}

Output:

Exception thrown: java.lang.ArithmeticException: / by zero

Exception thrown: java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 2

Out of the block

Task 7:

class task007exception {  
  
 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());  
 }  
 }  
}

output:

Inside fun().

Caught in main.demo exception by fun method

Process finished with exit code 0

Task8:

Task9:

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);

}

}

Output:

[1, 2, 3]

=== Code Execution Successful ===

Task 10

import java.util.\*;

class Main {

public static void main(String args[]){

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

al.add("Jaysree");

al.add("Hariharan");

System.out.println("Orignal 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);

}

}

Output:

Orignal List : [Jaysree, Hariharan]

After Adding element at index 1 : [Jaysree, Hello, Hariharan]

Element removed from index 0 : [Hello, Hariharan]

Element Prasunamba removed : [Hello, Hariharan]

List after updation of value : [K, Hariharan]

=== Code Execution Successful ===

Task 11:

What is the output of the below code snippet..  Explain ..

Code:

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);

}

}

Output:

15

=== Code Execution Successful ===

Task 12:

Use the above code and make the inner class as private and see the output..

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);

}

}

Output:

ERROR!

Main.java:11: error: OuterClass.InnerClass has private access in OuterClass

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

^

Main.java:11: error: OuterClass.InnerClass has private access in OuterClass

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

^

2 errors

Task 13:

Use the above code Task 011 and make the inner class static … see the output and explain..

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 = new OuterClass.InnerClass();

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

}

}

Output:

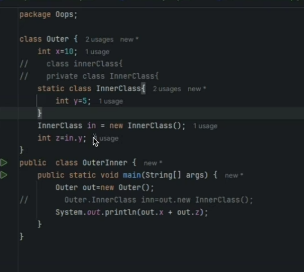
5,10

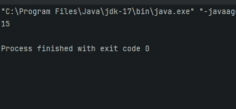
=== Code Execution Successful ===

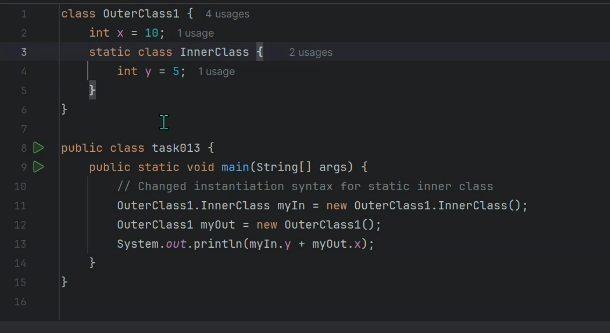
^

Task 14:

Use the above code Task 015 and create a method in inner class and return the outer class variable







Task 15:

Give reason or the code..

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);

}

}

Output:

5

=== Code Execution Successful ===

Task 17:

What are the features of Java 8?

1. Lambda Expressions
2. Functional Interfaces
3. Streams API
4. Default Methods
5. Method References
6. Java io improvements

Task 18:

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);

}

}

Output:

Iterator Value::0

Iterator Value::1

Iterator Value::2

Iterator Value::3

Iterator Value::4

Iterator Value::5

Iterator Value::6

Iterator Value::7

Iterator Value::8

Iterator Value::9

forEach anonymous class Value::0

forEach anonymous class Value::1

forEach anonymous class Value::2

forEach anonymous class Value::3

forEach anonymous class Value::4

forEach anonymous class Value::5

forEach anonymous class Value::6

forEach anonymous class Value::7

forEach anonymous class Value::8

forEach anonymous class Value::9

Consumer impl Value::0

Consumer impl Value::1

Consumer impl Value::2

Consumer impl Value::3

Consumer impl Value::4

Consumer impl Value::5

Consumer impl Value::6

Consumer impl Value::7

Consumer impl Value::8

Consumer impl Value::9

Task 20:

import java.util.\*;  
public class CollectionsDemo {  
  
 public static void main(String[] args) {  
 // ArrayList  
 List a1 = new ArrayList();  
 a1.add("Zara");  
 a1.add("Mahnaz");  
 a1.add("Ayan");  
 System.*out*.println(" ArrayList Elements");  
 System.*out*.print("\t" + a1);  
  
 // LinkedList  
 List l1 = new LinkedList();  
 l1.add("Zara");  
 l1.add("Mahnaz");  
 l1.add("Ayan");  
 System.*out*.println();  
 System.*out*.println(" LinkedList Elements");  
 System.*out*.print("\t" + l1);  
  
 // HashSet  
 Set s1 = new HashSet();  
 s1.add("Zara");  
 s1.add("Mahnaz");  
 s1.add("Ayan");  
 System.*out*.println();  
 System.*out*.println(" Set Elements");  
 System.*out*.print("\t" + s1);  
  
 // HashMap  
 Map m1 = new HashMap();  
 m1.put("Zara", "8");  
 m1.put("Mahnaz", "31");  
 m1.put("Ayan", "12");  
 m1.put("Daisy", "14");  
 System.*out*.println();  
 System.*out*.println(" Map Elements");  
 System.*out*.print("\t" + m1);  
 }  
}

output:

ArrayList Elements

[Zara, Mahnaz, Ayan]

LinkedList Elements

[Zara, Mahnaz, Ayan]

Set Elements

[Ayan, Zara, Mahnaz]

Map Elements

{Daisy=14, Ayan=12, Zara=8, Mahnaz=31}

Process finished with exit code 0