**Task 1**What do you understand by exceptions?

Answer:-Exceptions are errors that happen when a program is running (not when it is being written or compiled).

They stop the normal flow of a program and can cause it to crash if not handled properly.

**Task 2**  
 What are the categories of Exceptions do we have in Java? What are they?

Answer:-

**🧩 Java has 3 main categories of Exceptions:**

**1. Checked Exceptions**

✅ These are **checked at compile time**.  
If you don’t handle them, the program will **not compile**.

📌 Examples:

* IOException
* SQLException
* FileNotFoundException

🛠️ You must handle them using try-catch or throws.

try {

FileReader file = new FileReader("data.txt");

} catch (FileNotFoundException e) {

System.out.println("File not found!");

}

**2. Unchecked Exceptions (also called Runtime Exceptions)**

❌ These are **not checked at compile time**.  
They happen during program **execution**.

📌 Examples:

* ArithmeticException (e.g., divide by zero)
* NullPointerException (e.g., using a null object)
* ArrayIndexOutOfBoundsException

int a = 10 / 0; // ArithmeticException

You can handle them, but it’s **not required** by the compiler.

**3. Errors**

⚠️ These are **serious problems** that your program usually **cannot recover from**.

📌 Examples:

* OutOfMemoryError
* StackOverflowError

These are not exceptions you typically handle with try-catch.

**🔁 Summary Table:**

| **Category** | **Checked at Compile Time?** | **Can be Handled?** | **Examples** |
| --- | --- | --- | --- |
| Checked Exceptions | ✅ Yes | ✅ Yes | IOException, SQLException |
| Unchecked Exceptions | ❌ No | ✅ Optional | ArithmeticException, NullPointerException |
| Errors | ❌ No | ⚠️ Not Recommended | OutOfMemoryError, StackOverflowError |

**Task 3**  
Can you try the below code snippet and let me know which kind of exception is this ?

What is the output of the code..

// Java program to demonstrates handling

// the exception using try-catch block

import java.io.\*;

class Geeks {

    public static void main(String[] args)

    {

        int n = 10;

        int m = 0;

        try {

            // Code that may throw an exception

            int ans = n / m;

            System.out.println("Answer: " + ans);

        }

        catch (ArithmeticException e) {

            // Handling the exception

            System.out.println(

                "Error: Division by zero is not allowed!");

        }

catch (ArithmeticException e) {

            // Handling the exception

            System.out.println(

                "Error: Division by zero is not allowed!");

        }

        finally {

            System.out.println(

                "Program continues after handling the exception.");

        }

    }

}

Answer:- **package** samplePackage8;

**import** java.io.\*;

**class** Geeks {

**public** **static** **void** main(String[] args)

{

**int** n = 10;

**int** m = 0;

**try** {

// Code that may throw an exception

**int** ans = n / m;

System.***out***.println("Answer: " + ans);

}

**catch** (ArithmeticException e) {

// Handling the exception

System.***out***.println(

"Error: Division by zero is not allowed!");

}

**finally** {

System.***out***.println(

"Program continues after handling the exception.");

}

}

}

Output:- Error: Division by zero is not allowed!

Program continues after handling the exception.

**Task 4**List of checked and unchecked exceptions.

Answer:-

**✅ Checked Exceptions (Compile-Time Exceptions)**

These must be handled using try-catch or throws — or your code won't compile.

**📋 Common Checked Exceptions:**

| **Exception Name** | **Description** |
| --- | --- |
| IOException | When input/output operation fails |
| FileNotFoundException | When a file is not found |
| SQLException | When database access fails |
| ClassNotFoundException | When a class is not found |
| InterruptedException | When a thread is interrupted |
| ParseException | When parsing fails (e.g. date parsing) |
| NoSuchMethodException | When a method is not found using reflection |
| InstantiationException | When trying to create an object of an abstract class or interface |
| InvocationTargetException | When a method called by reflection throws an exception |
| MalformedURLException | Invalid URL format |

**❌ Unchecked Exceptions (Runtime Exceptions)**

These occur during program execution. The compiler **doesn't force** you to handle them.

**📋 Common Unchecked Exceptions:**

| **Exception Name** | **Description** |
| --- | --- |
| ArithmeticException | Divide by zero, etc. |
| NullPointerException | Accessing object with null reference |
| ArrayIndexOutOfBoundsException | Accessing array with invalid index |
| StringIndexOutOfBoundsException | Invalid index in string operations |
| NumberFormatException | Parsing a string to a number fails |
| IllegalArgumentException | When an illegal argument is passed |
| IllegalStateException | Method called at wrong time or state |
| ClassCastException | Invalid type casting |
| UnsupportedOperationException | Operation not supported |
| NegativeArraySizeException | Array created with negative size |

**🔁 Summary Table:**

| **Category** | **Checked By Compiler?** | **Must Handle?** | **Examples** |
| --- | --- | --- | --- |
| **Checked** | ✅ Yes | ✅ Yes | IOException, SQLException |
| **Unchecked** | ❌ No | ❌ Optional | NullPointerException, ArithmeticException |

**Task 5   
 package** samplePackage8;

**import** java.io.\*;

**class** Geeks {

**public** **static** **void** main(String[] args) {

**try** {

**int** a[] = **new** **int**[2];

**int** b = 4;

**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");

}

}

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

Out of the block

**Task 6   
package** samplePackage8;

**import** java.io.\*;

**class** Geeks {

**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** (ArithmeticException e) {

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

}

**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");

}

}

Output:-  
ArithmeticException thrown :java.lang.ArithmeticException: / by zero

Out of the block

**Task 7   
package** samplePackage8;

**import** java.io.\*;

**class** Geeks {

**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 | ArithmeticException e) {

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

}

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

}

}

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

Out of the block

**Task 8  
package** samplePackage8;

**import** java.io.\*;

**class** Geeks {

**public** **static** **void** main(String[] args) {

**try** {

**int** a[] = **new** **int**[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 9**

**package** samplePackage8;

**import** java.io.\*;

**class** Geeks {

// Method that throws a checked exception

**static** **void** fun() **throws** IllegalAccessException {

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

**throw** **new** IllegalAccessException("demo");

}

// Method that throws an unchecked exception

**static** **void** method2() {

**int**[] arr = **new** **int**[2];

System.***out***.println(arr[5]); // This will throw ArrayIndexOutOfBoundsException

}

// Method that throws a checked exception

**static** **void** method3() **throws** FileNotFoundException {

FileReader file = **new** FileReader("non\_existing\_file.txt"); // This file doesn't exist

}

**public** **static** **void** main(String args[]) {

**try** {

*fun*(); // Throws IllegalAccessException

*method2*(); // Throws ArrayIndexOutOfBoundsException

*method3*(); // Throws FileNotFoundException

}

**catch** (IllegalAccessException e) {

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

}

**catch** (ArrayIndexOutOfBoundsException e) {

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

}

**catch** (FileNotFoundException e) {

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

}

System.***out***.println("End of main");

}

}

Output:- Inside fun()

Caught in main: java.lang.IllegalAccessException: demo

End of main   
  
  
**Task 10   
package** samplePackage8;

**import** java.util.\*;

**class** Geeks {

**public** **static** **void** main(String args[]) {

ArrayList<Integer> numbers = **new** ArrayList<>();

**for** (**int** i = 1; i <= 10; i++) {

numbers.add(i);

}

System.***out***.println("ArrayList elements:");

**for** (**int** num : numbers) {

System.***out***.println(num);

}

}

}

Output:- ArrayList elements:

1

2

3

4

5

6

7

8

9

10

**Task 11   
package** samplePackage8;

**import** java.util.\*;

**class** Geeks {

**public** **static** **void** main(String args[]) {

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

al.add("Ranga Srinivas");

al.add("Nikhil");

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("Ranga Srinivas");

System.***out***.println("Element Ranga Srinivas removed : " + al);

al.set(0, "K");

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

}

}

Output:-

Original List : [Ranga Srinivas, Nikhil]

After Adding element at index 1 : [Ranga Srinivas, Hello, Nikhil]

Element removed from index 0 : [Hello, Nikhil]

Element Ranga Srinivas removed : [Hello, Nikhil]

List after updation of value : [K, Nikhil]

Task 12   
  
class MyException extends Exception {

public MyException(String message) {

super(message);

}

}

public class setText {

public static void main(String args[]) {

try {

throw new MyException("This is a custom exception");

}

catch (MyException ex) {

System.out.println("Caught");

System.out.println(ex.getMessage());

}

}

}  
output:- Caught

This is a custom exception  
  
  
**Task 13   
package** samplePackage8;

**class** OuterClass {

**int** x = 10;

**class** InnerClass {

**public** **int** getOuterValue() {

**return** x;

}

}

}

**public** **class** Task001{

**public** **static** **void** main(String[] args) {

OuterClass outer = **new** OuterClass();

OuterClass.InnerClass inner = outer.**new** InnerClass();

**int** value = inner.getOuterValue();

System.***out***.println("Value from OuterClass: " + value);

}

}

Output:- Value from OuterClass: 10