Sr.no Date Topic Sign

03

Page No.
Date

Q.1) Explain Exception Handling in Java?

→ Exception handling in Java is a mechanism to handle runtime errors that can occur in a program.
→ It helps to ensure that the program continues to execute normally, even when an exception is thrown.
→ Exception handling is one of the most important feature of java programming that allows you to handle the runtime errors cause by exception.
→ Exceptions are abnormal conditions that occur during the excecution of a program, and they can be handled using a try-catch block.
→ A exception is an object which is generated from the exception class or one of its subclass An exception object represents an error that has occurred.
→ There are 2 types:
1) Compile time error
2) Run time error

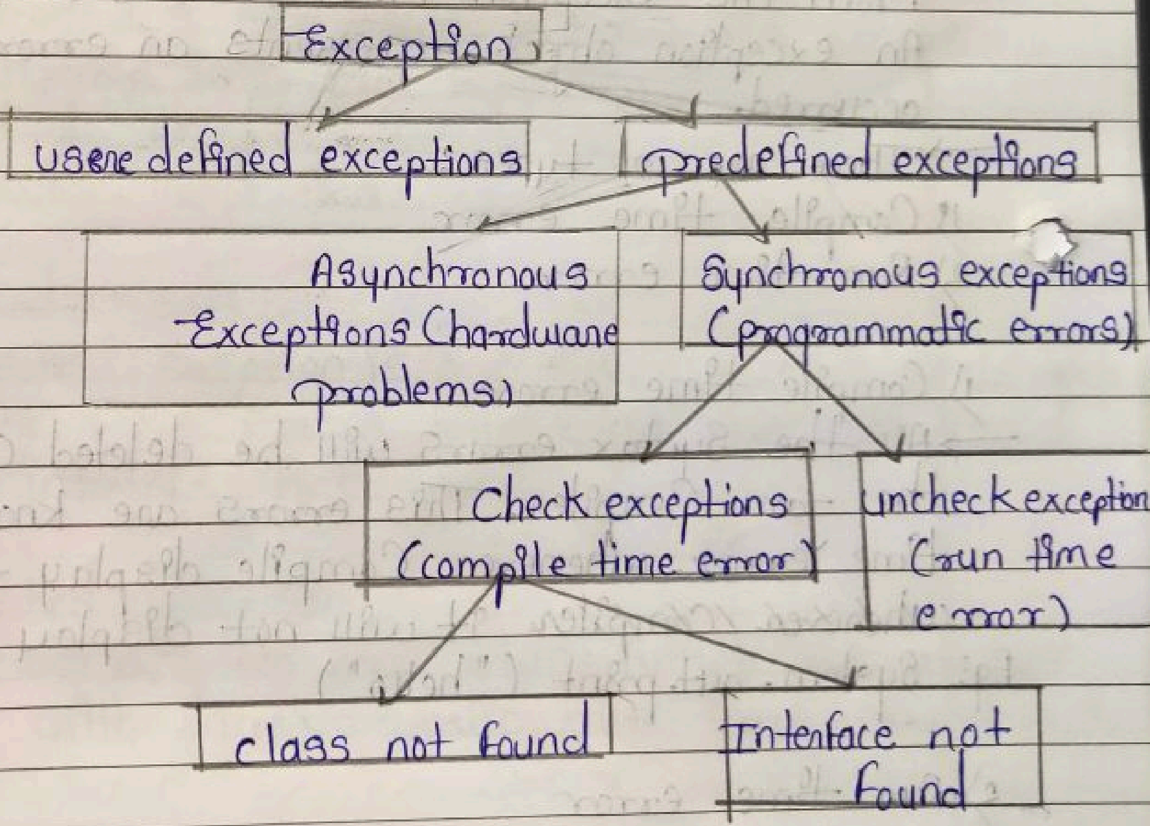
1) Compile time error
→ All the Syntax errors will be deleted can display by the java Compiler. This errors are known as Compile time error. whenever Compile display time errors, whenever /Compiler it will not display a file
Eg: System.out.print ("hello")

2) Run time error
→ Sometimes a program Compile Successfully Creating a

class file but many not run properly such program may produce wrong result and display and appropriate message to take corrective action. The task in known as exception handling to detect an exception circumstances.

Q.2> What are predefined exception in Java?

→ predefined exceptions are those which are develope d by sun micro system and supplied as a part of JDK to deal with universal problems.
→ Some of the universal problems are dividing by zero, invalid format of the number, invalid bounce of the array, etc


Exception
usere defined exceptions
predefined exceptions
Asynchronous Exceptions (hardware problems)
Synchronous exceptions (programmatic errors)
Check exceptions (compile time error)
uncheck exception (run time error)
class not found
Interface not found

→ exceptions are divided into two types. They are asynchronous exceptions and synchronous exceptions.

1) → Asynchronous exceptions are those which are always deals with hardware problems.

→ In order to deal with asynchronous exception there is a predefined class called Java.lang.Error. Java.lang.Error class is the super class for all asynchronous exception.

2) Synchronous exceptions are one which always deals with programmatic errors.
In order to deal with synchronous exceptions we must use a predefined class called Java.lang. Exception class.
Java.lang.Exception is the super class for all synchronous exceptions. Synchronous exceptions are divided into two types.

i) checked exception
ii) unchecked exception

i) checked exception
→ A checked exception is one which always deals with compile time errors regarding class not found and interface not found.

ii) unchecked exception
→ unchecked exception are those which are always deals with programmatic run time errors such as ArithmeticException, numberformatException, ArrayIndexoutofBoundsException, etc.

Date

Q.3) Explain try, Catch and finally block?

→ The 'try', 'catch', and 'finally' blocks are used in exception handling in programming

1) Try block

→ A try block is used to enclose the code that might throw an exception.

→ If an exception occurs within the try block, the rest of the code within the try block will not be executed.

2) Catch block

→ If an exception is thrown in the try block, it will be caught by the associated catch block.

→ The catch block contains the code that will be executed in response to the exception.

→ The catch block takes an argument, which is the type of exception that it will catch.

3) finally block

→ The finally block is optional and is executed after the 'try' and 'catch' blocks.

→ The code within the 'finally' block is guaranteed to be executed, whether an exception is throw or not.

→ This block is used to clean up resources or perform other tasks that must be done regardless of whether an exception is thrown or not.

• Example

→
```
try
{
    int a = 10/0 ;
}
catch (ArithmeticException e)
{
    System.out.println ("Exception Caught : Division
                          by zero.");
}
finally
{
    System.out.println ("finally block executed.");
}
```

output :

Q.4) Define use of multiple catch statement with example?

→ Multiple catch statements allow you to catch multiple exceptions in a single try-catch block.

→ This can be useful when you want to handle different exceptions in different ways or take different actions based on the type of exception that was thrown.

→ Here's an example of how you might use multiple Catch statements :

```
try
{
      // Code that might throw an exception
}
catch (FileNotFoundException e)
{
      // handle the FileNotFoundException
}
Catch (IOException e)
{
      // handle the IOException
}
Catch (Exception e)
{
      // handle any other exceptions
}
```

In this example, the code in the try block might throw an exception of type 'FileNotFoundException', 'IOException', or 'Exception'. If a 'FileNotFoundException' is thrown, the first catch block will be executed.
If an 'IOException' is thrown, the second catch blo
will be executed.
If any other exception is thrown, the third catch
block will be executed.

Q.5) What are advantages of exception handling?

→ 1) Improved Error Management

→ Exception handling provides a structured and uniform way to handle errors and prevents the program from abrupt termination.

2) Readability

→ Exception handling improves the readability of code by separating the error-handling logic from the normal code

3) Debugging

→ Exception handling makes it easier to debug the Code as the error messages are more meaningful and descriptive.

4) Reusability

→ Exception handling promotes the reuse of code by separating the error-handling logic from the main code.

5) Flexibility

→ Exception handling makes it possible to handling makes it possible to handle different types of exceptions in different ways, increasing the flexibility of the code

6) Separation of Concerns

→ Exception handling separates the error handling logic from the main code, leading to a clearer separation of concerns and improved maintainability.

Q.6) Write a shote note on throw statement.

→ The "throw" statement is used in programming to raise an exception in the code.

→ It stops the normal flow of execution and transfers control to an exception handler.

→ The argument passed to the throw statement can be any value, including an object that represents an error message.

→ The throw statement is often used to indicate a specific error condition, such as an invalid argument, and to allow the caller to take appropriate action to handle the error.

→ Java throw keyword is used to throw on exception in the core inside the function or the block of code.

→ Throw is used within the method.

→ The throw keyword is followed by an instance of exception to be thrown.

Q.7) Explain user define exception

→ user-defined exceptions, also known as custom exceptions, are exceptions that are created by the programmer to handle specific error conditions in the code.

→ Sometimes a program [illegible]

→ They are defined derived from the built-in exception classes in the programming language, and they allow you to customize the error handling process to match the needs of your application.

→ By defining custom exceptions you can provide more meaningful error messages and separate error handling logic for different error conditions.

→ This can make it easier to locate and debug issues in the code, as well as improve the overall user experience by providing a more appropriate response to errors.

→ For example, if you have a program that performs calculations, you might define a custom exception to indicate that a division by zero error has occurred, and throw this exception whenever a division operation results in a divide by zero scenario.

→ This allows you to handle this specific error condition in a different way than other types of exceptions, such as a file not found error, for example.