**Java Assignment5**

**Q1.What is Exception in Java?**

**Answer: Exception** is an unwanted or unexpected event, which occurs during the execution of a program, i.e. at run time, that disrupts the normal flow of the program’s instructions. Exceptions can be caught and handled by the program. When an exception occurs within a method, it creates an object. This object is called the exception object. It contains information about the exception, such as the name and description of the exception and the state of the program when the exception occurred.

**Q2.What is Exception Handling?**

**Answer: Exception Handling** in Java is one of the effective means to handle the runtime errors so that the regular flow of the application can be preserved. Java Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

**Q3.What is the difference between Checked and Unchecked Exceptions and Error?**

**Answer:**

**Checked Exception Unchecked Exception**

|  |  |
| --- | --- |
| Checked exceptions occur at compile time. | Unchecked exceptions occur at runtime. |
| The compiler checks a checked exception. | The compiler does not check these types of exceptions. |
| These types of exceptions can be handled at the time of compilation. | These types of exceptions cannot be a catch or handle at the time of compilation, because they get generated by the mistakes in the program. |
| They are the sub-class of the exception class. | They are runtime exceptions and hence are not a part of the Exception class. |
| Here, the JVM needs the exception to catch and handle. | Here, the JVM does not require the exception to catch and handle. |
| Examples of Checked exceptions:   * File Not Found Exception * No Such Field Exception * Interrupted Exception * No Such Method Exception * Class Not Found Exception | Examples of Unchecked Exceptions:   * No Such Element Exception * Undeclared Throwable Exception * Empty Stack Exception * Arithmetic Exception * Null Pointer Exception * Array Index Out of Bounds Exception * Security Exception |

**Q4.What are the difference between throw and throws in Java?**

**Answer:**

**throw**  **throws**

|  |  |
| --- | --- |
| The **throw** keyword is used inside a function. It is used when it is required to throw an Exception logically. | The **throws** keyword is used in the function signature. It is used when the function has some statements that can lead to exceptions. |
| The **throw** keyword is used to throw an exception explicitly. It can throw only one exception at a time. | The **throws** keyword can be used to declare multiple exceptions, separated by a comma. Whichever exception occurs, if matched with the declared ones, is thrown automatically then. |
| Syntax of **throw** keyword includes the instance of the Exception to be thrown. Syntax wise throw keyword is followed by the instance variable. | Syntax of **throws** keyword includes the class names of the Exceptions to be thrown. Syntax wise throws keyword is followed by exception class names. |
| **throw** keyword cannot propagate checked exceptions. It is only used to propagate the unchecked Exceptions that are not checked using the throws keyword. | **throws** keyword is used to propagate the checked Exceptions only. |

**Q5.What is multithreading in Java? mention its advantages**

**Answer:**  Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

This concurrent activity speeds applications up - one of the main benefits of multithreading.

MultiThreading allows both the full exploitation of parallel hardware and the effective use of multiple processor subsystems. While MT is essential for taking advantage of the performance of symmetric multiprocessors, it also provides performance benefits on uniprocessor systems by improving the overlap of operations such as computation and I/O.

Some of the most important benefits of MT are:

* Improved throughput. Many concurrent compute operations and I/O requests within a single process.
* Simultaneous and fully symmetric use of multiple processors for computation and I/O
* Superior application responsiveness. If a request can be launched on its own thread, applications do not freeze or show the "hourglass". An entire application will not block, or otherwise wait, pending the completion of another request.
* Improved server responsiveness. Large or complex requests or slow clients don't block other requests for service. The overall throughput of the server is much greater.
* Minimized system resource usage. Threads impose minimal impact on system resources. Threads require less overhead to create, maintain, and manage than a traditional process.
* Program structure simplification. Threads can be used to simplify the structure of complex applications, such as server-class and multimedia applications. Simple routines can be written for each activity, making complex programs easier to design and code, and more adaptive to a wide variation in user demands.
* Better communication. Thread synchronization functions can be used to provide enhanced process-to-process communication. In addition, sharing large amounts of data through separate threads of execution within the same address space provides extremely high-bandwidth, low-latency communication between separate tasks within an application.

**Q6.Write a program to create and call a custom exception**

**Answer:** // A Class that represents use-defined exception

class MyException extends Exception {

public MyException(String s)

{

// Call constructor of parent Exception

super(s);

}

}

// A Class that uses above MyException

public class Test {

public static void main(String args[])

{

try {

// Throw an object of user defined exception

throw new MyException("UserException");

}

catch (MyException ex) {

System.out.println("Caught");

// Print the message from MyException object

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

}

}

}

**Q7.How can you handle exceptions in Java?**

**Answer:** Java provides specific keywords for exception handling purposes.

1. **throw** – We know that if an error occurs, an exception object is getting created and then Java runtime starts processing to handle them. Sometimes we might want to generate exceptions explicitly in our code. For example, in a user authentication program, we should throw exceptions to clients if the password is null. The throw keyword is used to throw exceptions to the runtime to handle it.
2. **throws** – When we are throwing an exception in a method and not handling it, then we have to use the throws keyword in the method signature to let the caller program know the exceptions that might be thrown by the method. The caller method might handle these exceptions or propagate them to its caller method using the throws keyword. We can provide multiple exceptions in the throws clause, and it can be used with the main() method also.
3. **try-catch** – We use the try-catch block for exception handling in our code. try is the start of the block and  is at the end of the try block to handle the exceptions. We can have multiple catch  blocks with a try block. The try-catch block can be nested too. The catch block requires a parameter that should be of type Exception .
4. **finally** – the finally block is optional and can be used only with a try-catch block. Since exception halts the process of execution, we might have some resources open that will not get closed, so we can use the finally block. The finally block always gets executed, whether an exception occurred or not.

**Q8.What is Thread in Java?**

**Answer:** A **Thread** is a very light-weighted process, or we can say the smallest part of the process that allows a program to operate more efficiently by running multiple tasks simultaneously.

Separate flow of execution is called "Thread".

if there is only one flow then it is called "SingleThread" programming.

For every thread there would be a separate job.

**Q9. What are the two ways of implementing thread in Java?**

**Answer:** In java we can define a thread in 2 ways

a. Implementing Runnable interface

b. extending Thread class

**Q10.What do you mean by garbage collection?**

**Answer:** Garbage collection in Java is the automated process of deleting code that's no longer needed or used. This automatically frees up memory space and ideally makes coding Java apps easier for developers. Java applications are compiled into bytecode that may be executed by a JVM.