5

Q1. What is an Exception in Java?
In Java, an exception is an event that occurs during the execution of a program and disrupts the normal flow of instructions. It represents an abnormal condition or error situation that may occur while the program is running.

Q2. What is Exception Handling?
Exception handling in Java is a mechanism to handle and recover from exceptional situations that may occur during the execution of a program. It allows you to catch and handle exceptions, preventing them from causing the program to terminate abruptly.
Exception handling involves the use of try-catch blocks. The code that may throw an exception is enclosed within the try block, and any potential exceptions are caught and handled in the catch block. Additionally, exception handling also provides the ability to perform cleanup operations using the finally block.

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

- Checked Exceptions: Checked exceptions are the exceptions that are checked at compile-time by the Java compiler. These exceptions are subclasses of Exception but not subclasses of RuntimeException. Examples of checked exceptions include IOException, SQLException, and ClassNotFoundException. Code that can potentially throw checked exceptions must handle them using try-catch blocks or declare them in the method signature using the throws keyword.
- Unchecked Exceptions: Unchecked exceptions are exceptions that are not checked at compile-time by the Java compiler. These exceptions are subclasses of RuntimeException. Examples of unchecked exceptions include NullPointerException, ArrayIndexOutOfBoundsException, and ArithmeticException. Unlike checked exceptions, code is not required to handle or declare unchecked exceptions explicitly.

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

- throw is used to explicitly throw an exception within a method. It is followed by an instance of an exception class or subclass. When the throw statement is executed, it terminates the execution of the current method and passes the control to the nearest enclosing try-catch block or the caller of the method.

- throws is used in the method signature to declare that the method may throw one or more exceptions. It is followed by the names of the exceptions separated by commas. By declaring the exceptions with throws, the method indicates that it may produce exceptions that need to be handled by the caller of the method or the next higher-level exception handler.

Q5. What is multithreading in Java? Mention its advantages.

Multithreading in Java refers to the concurrent execution of multiple threads within a single program. A thread is a lightweight subprocess that can perform tasks independently and concurrently with other threads.

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

```
// Custom Exception Class
class MyException extends Exception {
   public MyException(String message) {
      super(message);
   }
}
```

```
// Main Class
public class CustomExceptionExample {
   public static void main(String[] args) {
      try {
         // Throw custom exception
         throw new MyException("This is a custom exception");
      } catch (MyException e) {
         // Catch and handle the custom exception
         System.out.println("Caught Exception: " + e.getMessage());
      }
   }
}
```

Q7. How can you handle exceptions in Java?

Exceptions in Java can be handled using the following mechanisms:

1. try-catch Blocks: The code that may throw an exception is enclosed within a try block, and the potential exceptions are caught and handled in one or more catch blocks. The catch block specifies the type of exception it can handle and contains the code to handle the exception.
2. finally Block: The finally block is used to specify code that should be executed regardless of whether an exception occurred or not. It is typically used for cleanup operations, such as closing resources, that need to be executed even if an exception is thrown.
3. Multiple catch Blocks: Multiple catch blocks can be used to handle different types of exceptions separately. Each catch block handles a specific exception type and contains the code to handle that particular exception.
4. throw Statement: The throw statement is used to manually throw an exception. It allows you to create and throw custom exceptions or throw predefined exceptions to indicate exceptional conditions.
5. try-with-resources: In Java 7 and later versions, the try-with-resources statement is used to automatically close resources that implement the AutoCloseable interface. It eliminates the need for explicit resource cleanup in the finally block.

Q8. What is a Thread in Java?

A thread in Java represents a sequential flow of control within a program. It is a lightweight subprocess that can execute independently and concurrently with other threads. Each thread has its own call stack and program counter, allowing it to execute its own set of instructions.

Q9. What are the two ways of implementing threads in Java?
In Java, there are two ways to implement threads:

1. Extending the Thread Class

```
class MyThread extends Thread {
   public void run() {
      // Code to be executed in the new thread
   }
}

// Create and start the thread
MyThread thread = new MyThread();
thread.start();
```

2. Implementing the Runnable Interface:

```
class MyRunnable implements Runnable {
    public void run() {
        // Code to be executed in the new thread
    }
}

// Create the runnable object
MyRunnable runnable = new MyRunnable();

// Create the thread and start it
Thread thread = new Thread(runnable);
thread.start();
```

Q10. What do you mean by garbage collection?

Garbage collection in Java is the automatic process of reclaiming memory occupied by objects that are no longer in use. It is a mechanism provided by the Java Virtual Machine (JVM) to manage memory allocation and deallocation.

In Java, objects are dynamically allocated on the heap memory. The garbage collector periodically identifies and releases the memory occupied by objects that are no longer reachable or referenced by any part of the program. It frees up memory resources and eliminates the need for manual memory management.