Assignment Questions 5 (Java)

Q1.What is Exception in Java?

Ans:In Java, an exception is an event that occurs during the execution of a program and disrupts the normal flow of the program. It represents an abnormal condition or error that may occur during runtime. Exceptions can occur due to various reasons such as invalid input, resource unavailability, or programming errors. When an exception occurs, it is thrown by the code that encounters the exceptional condition and can be caught and handled by appropriate exception handling mechanisms in the program. Java provides a robust exception handling mechanism to handle and recover from such exceptional situations.

Q2.What is Exception Handling?

Ans:Exception handling in Java is a mechanism that allows programmers to handle and manage exceptions that occur during program execution. It involves catching and handling exceptions to prevent program termination and provide appropriate actions or error messages. By using try-catch blocks, exceptions can be caught and specific code can be executed to handle the exception. Additionally, the use of the "finally" block ensures that certain code is executed regardless of whether an exception occurred or not. Exception handling helps improve program reliability and provides a structured approach to deal with exceptional situations.

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

Ans: In Java, checked exceptions are the exceptions that are checked at compile-time and must be handled explicitly using try-catch blocks or declared in the method signature. Examples include IOException and ClassNotFoundException. On the other hand, unchecked exceptions are not checked at compile-time and do not require explicit handling. Examples include NullPointerException and ArrayIndexOutOfBoundsException. Errors, such as OutOfMemoryError or StackOverflowError, are exceptional conditions that usually cannot be recovered from and are not meant to be caught or handled programmatically. They indicate serious problems that may cause the application to terminate.

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

Ans:In Java, "throw" is used to explicitly throw an exception within a method. It is used when a specific exceptional condition is encountered and needs to be handled. On the other hand, "throws" is used in method signatures to declare that the method may potentially throw a specific type of exception. It is used to delegate the responsibility of handling the exception to the calling method or propagate the exception further up the call stack. "throw" is used within a method, while "throws" is used in the method declaration to indicate possible exceptions.

Q5.What is multithreading in Java? mention its advantages

Ans:Multithreading in Java refers to the concurrent execution of multiple threads within a single program. It allows different parts of a program to run concurrently, improving performance and responsiveness. Advantages of multithreading include increased efficiency by utilizing multiple CPU cores, enhanced program responsiveness by handling multiple tasks simultaneously, better resource utilization, improved scalability, and the ability to perform background tasks while the main thread continues with other operations. Multithreading also enables concurrent access to shared resources and supports parallel computing for computationally intensive tasks.

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

Ans:

```
class MyCustomException extends Exception {
    public MyCustomException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        try {
            int age = -5;
            if (age < 0) {
                throw new MyCustomException("Age cannot be negative");
            }
        } catch (MyCustomException e) {
            System.out.println("Custom Exception Caught: " + e.getMessage());
        }
    }
}
```

Q7.How can you handle exceptions in Java?

Ans:In Java, exceptions can be handled using the try-catch-finally construct. The code that may throw an exception is enclosed within the try block. If an exception occurs, it is caught by a catch block that matches the exception type. Multiple catch blocks can be used to handle different types of exceptions. The finally block is optional and is used to execute code that should always run, regardless of whether an exception occurred or not. Additionally, exceptions can be propagated using the throws keyword to delegate handling to the caller method.

Q8.What is Thread in Java?

Ans: In Java, a thread is a lightweight unit of execution within a program. It represents an independent path of execution that can run concurrently with other threads. Threads allow for concurrent and parallel processing, enabling multiple tasks to be executed simultaneously. Each thread has its own call stack and can execute code independently. Threads can be created either by extending the Thread class or implementing the Runnable interface. By utilizing threads, Java applications can achieve concurrent execution, improve performance, and handle multiple tasks efficiently.

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

1. By extending the `Thread` class: This involves creating a new class that extends the `Thread` class and overrides its `run()` method to define the code that the thread will execute. The class can then be instantiated and the thread started using the `start()` method.

2. By implementing the `Runnable` interface: This involves creating a new class that implements the `Runnable` interface and overrides its `run()` method. The class is then passed as an argument to a `Thread` constructor or assigned to a `Thread` object. The thread is started using the `start()` method of the `Thread` object.

Both approaches allow for the creation and execution of threads in Java programs.

Q10.What do you mean by garbage collection?

Ans:Garbage collection in Java is an automatic memory management process where the JVM (Java Virtual Machine) automatically reclaims memory occupied by objects that are no longer referenced by the program. It identifies and deallocates memory that is no longer in use, freeing up system resources. The garbage collector traces objects and determines which ones are still reachable, releasing the memory of unreferenced objects. Garbage collection relieves programmers from manual memory management tasks, improves memory efficiency, and helps prevent memory leaks and memory-related bugs.