Assignment 5

Q1.What is Exception in Java?

Ans) An exception in Java is an event that occurs during the execution of a program that disrupts the normal flow of the program's instructions. Exceptions can be caused by a variety of factors, such as:

- Runtime exception
- Logical error
- User input error

Q2.What is Exception Handling?

Ans)Exception handling is a mechanism in programming languages, including Java, that allows you to deal with and respond to exceptions or errors that occur during the execution of a program. It provides a structured way to handle exceptional situations and control the flow of the program when an unexpected condition arises.

In other word it is gracefully termination of program.

It can be handled using “try catch” statement

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

Ans)**Checked Exceptions:**

1. Checked exceptions are the exceptions that are checked at compile-time.
2. They are subclasses of Exception but not of **“RuntimeException”**.
3. Checked exceptions must be declared in the method signature or handled using try-catch blocks.

4. Examples of checked exceptions include `**IOException**`, “SQLException”, and “ClassNotFoundException”.

Unchecked Exceptions:

1. Unchecked exceptions are the exceptions that are not checked at compile-time.
2. They are subclasses of ‘**RuntimeException**’.
3. Unchecked exceptions do not require explicit declaration or handling.
4. Examples of unchecked exceptions include “**NullPointerException**”, “**ArrayIndexOutOfBoundsException**”, and “**IllegalArgumentException**”.

Error:

1. Errors are exceptional conditions that typically occur due to a system-level error or a resource exhaustion.
2. Errors are subclasses of `Error`.
3. Errors are generally not recoverable, and it is not expected to handle them explicitly in the code.
4. Examples of errors include “**OutOfMemoryError**”, “**StackOverflowError**”, and “**VirtualMachineError**”.

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

Ans)In Java, "throw" and "throws" are two keywords used in exception handling, but they serve different purposes:

1. throw:
   - "throw" is used to explicitly throw an exception from a method or block of code.
   - It is followed by an instance of an exception class or a subclass of Throwable.
   - It is used to indicate that an exceptional condition has occurred within the code.
   - The throw statement terminates the normal flow of the program and transfers control to the nearest enclosing try-catch block or method that can handle the thrown exception.

   ⇒ Example: **“throw new NullPointerException()”**

2. throws:
   - "throws" is used in a method signature to declare that the method may throw one or more types of exceptions.
   - It specifies the exceptions that a method might throw, allowing the calling code to handle them appropriately.
   - Multiple exceptions can be declared using a comma-separated list.
   - It does not throw an exception itself but rather provides information to the caller about the possible exceptions that can occur within the method.

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. A thread is a lightweight unit of execution that represents an independent flow of control.

⇒Advantages of multithreading in Java:

1. Increased Responsiveness: It enables concurrent execution of multiple threads, ensuring that the user interface remains interactive and doesn't freeze.

2. Improved Performance: By utilising multiple threads, a program can leverage the available CPU resources effectively, leading to improved performance and faster execution.

3. Resource Sharing: Threads within the same process can share resources, such as memory, files, and network connections, without the need for duplicating them. This allows for efficient utilisation of system resources.

4. Enhanced Concurrency: Multithreading enables the execution of multiple tasks concurrently, promoting parallelism and allowing for efficient handling of multiple operations simultaneously.

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

Ans)

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

class CustomExceptionExample {
public static void main(String[] args) {
try {
processInput(10);
} catch (MyCustomException e) {
System.out.println("Custom Exception Caught: " +
e.getMessage());
}
}
public static void processInput(int value) throws
MyCustomException {
if (value < 0) {
throw new MyCustomException("Input value cannot be
negative");
} else {
System.out.println("Input value is valid");
}
}
}
```

Q7.How can you handle exceptions in Java?

Ans)There are two main ways to handle exceptions in Java:

- The try-catch block: This is the most common way to handle exceptions.The try-catch block allows us to specify a block of code that should be executed and a catch block that should be executed if an exception is thrown
- The throws clause: The throws clause allows you to specify the exceptions that a method or constructor can throw.This allows the caller of the method or constructor to be prepared to handle the exceptions.

Q8.What is Thread in Java?

Ans)In Java, a thread is a lightweight unit of execution that represents an independent path of execution within a program. It allows multiple tasks or parts of a program to run concurrently, enabling parallelism and efficient utilisation of system resources. Threads can be created and managed using the “Thread” class or by implementing the “Runnable” interface. Each thread has its own call stack and can execute code independently. Multithreading in Java allows for concurrent execution, responsiveness, and improved performance by dividing tasks into smaller units of execution that can run simultaneously, leading to faster execution and enhanced program functionality.

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

Ans)In Java, there are two ways to implement threads:

1. Extending the Thread class:
   - You can create a thread by extending the “Thread” class and overriding its “run()” method, which contains the code to be executed in the thread.
   - Here's an example of creating a thread by extending the “Thread” class:

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

   - To start the thread, create an instance of the subclass and call the `start()` method:

   ```
   MyThread myThread = new MyThread();
   myThread.start();
   ```

2. Implementing the Runnable interface:
   - Another way to create a thread is by implementing the “Runnable” interface, which requires implementing the “run()” method.
   - Here's an example of creating a thread by implementing the “Runnable” interface:

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

- To start the thread, create an instance of the “Runnable” implementation and pass it to the “Thread” constructor. Then, call the “start()” method on the “Thread” instance:

```
MyRunnable myRunnable = new MyRunnable();
Thread thread = new Thread(myRunnable);
thread.start();
```

Q10.What do you mean by garbage collection?

Ans)Garbage collection in programming refers to the automatic memory management process performed by the runtime environment to reclaim memory that is no longer in use by the program. In languages like Java, where memory allocation and deallocation are handled by the runtime, garbage collection plays a vital role in memory management. It identifies and frees up memory occupied by objects that are no longer reachable or referenced by the program, thus preventing memory leaks and improving resource utilisation. The garbage collector scans the heap, identifies unused objects, and releases their memory, allowing it to be reused for future allocations. This automated process relieves the developer from manual memory management and helps in ensuring memory efficiency and stability.