**Task – 11 JAVA**

1. **What are the four access modifiers available in Java and what is their significance in terms of class, method and variable accessibility?**

In Java, there are four access modifiers:-

1. **Public:** Public access modifier allows a class, method, or variable to be accessible from any other class in the same package or from any other package.
2. **Protected:** Protected access modifier allows a class, method, or variable to be accessible within its own package or by subclasses (classes that extend the class) in other packages.
3. **Default (no modifier):** Default access modifier (also known as package-private) allows a class, method, or variable to be accessible only within its own package.
4. **Private:** Private access modifier restricts the accessibility of a method or variable to within its own class only. It's not accessible outside the class, not even in subclasses.

Here's a summary of their significance:

* **Class:**
  + Public: Can be accessed from anywhere.
  + Protected, default, private: Cannot be accessed from outside the class (except nested classes where private can be accessed).
* **Method:**
  + Public: Can be accessed from anywhere.
  + Protected: Can be accessed within the same package or by subclasses.
  + Default: Can be accessed only within the same package.
  + Private: Can be accessed only within the same class.
* **Variable:**
  + Public, protected, default: Accessibility depends on where the variable is declared (class level, method level, etc.) and follows the same rules as methods.
  + Private: Can only be accessed within the same class.

These modifiers provide control over the visibility and accessibility of classes, methods, and variables, helping to enforce encapsulation.

1. **What is the difference between Exception and Error?**

Exceptions and errors are both types of Throw able objects in Java, but they serve different purposes and occur under different circumstances:

1. **Exceptions:**
   * Exceptions represent conditions that generally can be caught and handled by the application.
   * They are typically caused by problems that a programmer should anticipate and handle, such as invalid user input, file not found, network connection issues, etc.
   * Exceptions are subclasses of java.lang.Exception and are checked at compile time, meaning that the code must either handle them with a try-catch block or declare that it throws the exception.
   * Examples include IOException, NullPointerException, ArrayIndexOutOfBoundsException, etc.
   * Exceptions are recoverable, meaning that the program may be able to continue execution after handling the exception.
2. **Errors:**
   * Errors represent serious problems that are typically beyond the control of the application.
   * They are often caused by external factors or issues with the environment in which the application is running, such as out of memory errors, stack overflow errors, hardware failures, etc.
   * Errors are subclasses of java.lang.Error and are unchecked, meaning that the compiler does not require them to be caught or declared.
   * Examples include OutOfMemoryError, StackOverflowError, NoClassDefFoundError, etc.
   * Errors are generally not recoverable, and attempting to handle them may not be appropriate or effective. They usually indicate severe problems that may require intervention at a higher level, such as restarting the application or fixing underlying system issues.

In summary, exceptions are used to handle expected and recoverable problems within the application logic, while errors indicate severe problems that are typically outside the scope of the application's control and may require more drastic measures to address.

1. **What is the difference between checked Exception and unchecked Exception?**

Checked exceptions and unchecked exceptions are both types of exceptions in Java, but they differ in how they are enforced by the compiler and how they are handled in the code:

1. **Checked Exceptions**:
   * Checked exceptions are exceptions that are checked at compile time by the Java compiler.
   * They are typically exceptions that a programmer should anticipate and handle, such as I/O exceptions, database access errors, etc.
   * Checked exceptions must either be caught using a try-catch block or declared in the method signature using the throws keyword.
   * Examples include IOException, SQLException, ClassNotFoundException, etc.
   * The compiler ensures that checked exceptions are either caught or declared to be thrown by the method. Failure to handle or declare checked exceptions results in a compilation error.
2. **Unchecked Exceptions:**
   * Unchecked exceptions, also known as runtime exceptions, are exceptions that are not checked by the compiler at compile time.
   * They are typically caused by programming errors or exceptional conditions that are not easily recoverable, such as null pointer dereference, array index out of bounds, arithmetic overflow, etc.
   * Unchecked exceptions do not need to be caught or declared using a try-catch block or the throws keyword.
   * Examples include NullPointerException, ArrayIndexOutOfBoundsException, ArithmeticException, etc.
   * Unchecked exceptions can occur at runtime, and they are not required to be explicitly handled by the programmer. However, it's generally good practice to anticipate and handle them where appropriate to prevent unexpected application crashes.

In summary, the main difference between checked and unchecked exceptions lies in how they are enforced by the compiler and how they are handled in the code. Checked exceptions must be caught or declared, whereas unchecked exceptions are not required to be caught or declared.