Task-11

QA.1

     In Java, there are four access modifiers that control the visibility and accessibility of classes, methods, and variables within a program:

1. public: When a class, method, or variable is declared with the `public` access modifier, it can be accessed from any other class in the same package or from any other package. This means that public members are accessible to all classes regardless of their location within the project.

2. protected: Members declared with the `protected` access modifier are accessible within the same package or by subclasses (regardless of whether they are in the same package or not). It means that they are visible to the class itself, its subclasses, and other classes in the same package.

3. default (no modifier): If no access modifier is specified, then by default, the member is accessible within the same package only. This means that classes, methods, or variables with default access can only be accessed by other classes within the same package.

4. private: Members declared with the `private` access modifier are accessible only within the same class. They cannot be accessed by any other class, even if they belong to the same package. This provides the highest level of encapsulation and is often used to hide implementation details and prevent direct access from outside the class.

Here's a summary of their significance in terms of class, method, and variable accessibility:

Class: Only `public` and `default` access modifiers can be applied to classes. A `public` class can be accessed from any other class in the same project, while a class with `default` access can only be accessed by other classes in the same package.

Method: All four access modifiers can be applied to methods. The choice of access modifier affects which classes can invoke the method. `public` methods are accessible from any class, `protected` methods are accessible from subclasses and classes in the same package, `default` methods are accessible from classes in the same package, and `private` methods are only accessible within the declaring class.

Variable: Like methods, all four access modifiers can be applied to variables. The access modifier determines which classes can access or modify the variable's value. `public` variables can be accessed from any class, `protected` variables can be accessed from subclasses and classes in the same package, `default` variables can be accessed from classes in the same package, and `private` variables are only accessible within the declaring class.

QA.2

     In Java, both exceptions and errors are types of Throwable objects, but they represent different kinds of problems that can occur during the execution of a program.

1. Exception:

   Exceptions are conditions that occur during the execution of a program that disrupt the normal flow of control.

   They are typically caused by the actions of the program or the environment in which it is running, and they are expected to be recoverable.

   Exceptions can be handled using try-catch blocks, allowing the program to respond to the exceptional condition and take appropriate action to recover from it.

   Examples of exceptions include FileNotFoundException, NullPointerException, and ArithmeticException.

2. Error:

    Errors, on the other hand, are abnormal conditions that occur during the execution of a program that typically indicate serious problems that cannot be easily handled or recovered from.

   Errors are often caused by problems with the environment in which the program is running, such as system resource limitations, hardware failures, or bugs in the Java Virtual Machine (JVM).

   Unlike exceptions, errors are not expected to be caught or handled by the program, and attempting to do so may not be feasible or advisable.

   Examples of errors include OutOfMemoryError, StackOverflowError, and VirtualMachineError.

         In  the  summary, exceptions represent exceptional conditions that occur during the normal execution of a program and are expected to be handled by the program, whereas errors represent serious problems that typically cannot be handled by the program and may require intervention at a higher level, such as by restarting the JVM or fixing underlying system issues.

QA.3

     In Java, exceptions are categorized into two main types: checked exceptions and unchecked exceptions. The difference between them lies in how they are handled by the compiler and the programmer:

1.  Checked Exceptions :

   Checked exceptions are subclasses of `Exception` (excluding `RuntimeException` and its subclasses) that the compiler requires to be handled explicitly by the programmer.

    When a method throws a checked exception, the method must either handle the exception using a try-catch block or declare the exception using the `throws` keyword in its method signature.

    The compiler checks for the handling of checked exceptions during compile-time, so if a checked exception is not handled properly, the code will not compile.

    Examples of checked exceptions include `IOException`, `SQLException`, and `ClassNotFoundException`.

2. Unchecked Exceptions :

    Unchecked exceptions are subclasses of `RuntimeException` and its subclasses, as well as `Error` and its subclasses.

    Unlike checked exceptions, unchecked exceptions are not required to be handled explicitly by the programmer.

    Unchecked exceptions typically represent programming errors or conditions that are beyond the control of the programmer, such as invalid arguments to a method or attempts to access elements beyond the bounds of an array.

    Since unchecked exceptions are not checked by the compiler at compile-time, the compiler does not enforce handling them using try-catch blocks or the `throws` keyword.

    Examples of unchecked exceptions include `NullPointerException`, `ArrayIndexOutOfBoundsException`, and `IllegalArgumentException`.

     In summary, the main difference between checked and unchecked exceptions in Java lies in how they are handled by the compiler and the programmer. Checked exceptions must be handled explicitly by the programmer, either through try-catch blocks or by declaring them in the method signature, while unchecked exceptions are not required to be handled explicitly and are typically used to represent programming errors or conditions beyond the programmer's control.