Assignments Solution for Week -1

Akash Pal - akash.majida@gmail.com

1. What are Primitive Data types in Java?
In Java, primitive data types are the basic building blocks for storing simple values. These include byte, short, int, long for integers; float, double for floating-point numbers; char for characters; and boolean for true/false values. These data types are essential for efficiently managing and manipulating fundamental types of data in Java programs.

2. What are the Identifiers in Java?
In Java, identifiers serve as names for various elements in a program, including variables, classes, methods, and labels. They must follow specific rules, such as starting with a letter, underscore, or dollar sign, and can consist of letters, digits, underscores, or currency characters. Identifiers are case-sensitive, distinguishing between uppercase and lowercase letters. While adhering to conventions like starting variables with lowercase letters and classes with uppercase letters, identifiers help uniquely identify and reference program components. Examples include myVariable, MyClass, and calculateTotal.

3. What is final class in Java?
In Java, a final class is one that cannot be extended or subclassed. When a class is declared as final, it signifies that its design is complete, and it is not intended for further modification or extension. This restriction prevents other classes from inheriting its properties and methods. The use of final classes is common in scenarios where the class's behavior should remain unchanged, providing design stability, and in situations where security or optimization considerations warrant the prevention of subclassing.

4. What are the two ways to make a class final?
In Java, there are two ways to make a class final:
Declare the Class as Final:
You can explicitly declare a class as final by using the final keyword in the class declaration. For example:
```
final class MyFinalClass {
    // Class members and methods
}
```
This approach ensures that no other class can extend or inherit from MyFinalClass.
Use the final Modifier on Class Members:
Alternatively, even if a class itself is not declared as final, you can use the final modifier on specific class members (fields or methods) to prevent them from being overridden or modified in subclasses. For example:
```
class MyClass {
    final int myFinalField = 42;

    final void myFinalMethod() {
        // Method implementation
    }
}
```
In this case, the myFinalField and myFinalMethod cannot be overridden or modified in any subclass of MyClass

5. Can we create an instance of final class in another class?
No, We cannot create an instance of a final class in another class by using the new keyword to instantiate it. The whole purpose of declaring a class as final is to prevent it from being subclassed or extended, and as a consequence, instances of that class cannot be created by any other class.

6. What is Volatile keyword used for ?
In Java, the volatile keyword is used to indicate that a variable's value may be changed by multiple threads simultaneously. Specifically, it ensures that any thread reading the variable sees the most recent modification made by any other thread. This is important in multithreaded programming where one thread may update a variable, and other threads need to be aware of the changes.

When a variable is declared as volatile, the compiler and the Java Virtual Machine (JVM) are informed that the variable can be changed by multiple threads and that certain optimizations should be avoided to maintain thread safety.

7. What is the use of Transient Keyword?
In Java, the transient keyword is used to indicate that a variable should not be serialized when the object containing it is serialized. Serialization is the process of converting an object into a stream of bytes, typically for the purpose of saving the object's state to a file or sending it across a network.

When a variable is marked as transient, it tells the Java serialization mechanism to skip that variable during the serialization process. This can be useful in situations where certain variables should not be persisted or transmitted with the object.

8. What are types of casting in Java?
There are two types of casting in Java
Primitive Casting: When the data is casted from one primitive type (like int, float, double etc... to another primitive type, then it is called primitive casting.
Derived Casting : When the data is casted from one derived type to another derived type, then it is called derived casting.

9. What is boxing and Unboxing?
Boxing and unboxing are operations in Java that involve the conversion between primitive data types and their corresponding wrapper classes.

Boxing:
Boxing is the process of converting a primitive data type into its equivalent wrapper class.
For example, converting an int to an Integer.
In modern Java, autoboxing allows for automatic conversion between primitive types and their corresponding wrapper classes without the need for explicit method calls.

Unboxing:
Unboxing is the process of converting an object of a wrapper class back to its corresponding primitive data type.
For example, converting an Integer back to an int.
Similar to autoboxing, modern Java supports autounboxing, which allows for automatic conversion from wrapper classes to primitive types.

10. What is the difference between keywords, identifiers and literals in java ?
In Java, and in programming languages in general, keywords, identifiers, and literals serve distinct roles in the structure and expression of code.

Keywords:
Keywords are reserved words in a programming language that have a predefined meaning and cannot be used as identifiers (names for variables, classes, etc.).
In Java, examples of keywords include class, int, if, else, for, while, return, and others.

Keywords are an integral part of the language syntax and have specific functionality associated with them.

Identifiers:
Identifiers are names given to various elements in a program, such as variables, classes, methods, and labels.
Identifiers must follow certain rules, like starting with a letter, underscore, or dollar sign, and can include letters, digits, underscores, or currency characters.
Unlike keywords, identifiers are user-defined, and they play a crucial role in making code readable and maintainable.
Examples of identifiers include variable names (myVariable), class names (MyClass), and method names (calculateTotal).

Literals:
Literals represent fixed values in the source code. They are used to express data directly in the code.
In Java, common types of literals include numeric literals (e.g., 42, 3.14), string literals (e.g., "Hello"), character literals (e.g., 'A'), boolean literals (true or false), and null literal (null).
Literals are used to initialize variables or provide values directly in expressions.