**NAME: NDIRANGU JOANNE NYOKABI**

**REG NO: 23/02370**

**Assignment 1(Theory questions)**

1. **Differences between primitive and reference data types**

Primitive data types are data types that store an actual value directly, making the memory faster and more efficient to access. Examples are Boolean, string, char, integer, etc. while Reference data types that store multiple key-value pairs. I.e. class, arrays, etc.

1. **Define the scope of a variable**

Local variables are defined within a method, constructor, or block and are only accessible within that method, constructor, or block.

Global variables can be interpreted to mean instance variables and class variables, these are broader than local variables.

**Instance variables**

These are declared within a class but outside any method, constructor, or block. They are tied to instances of the class(objects).

**Scope:** They are accessible by all methods, constructors, and blocks within the class. They can also be accessed from outside the class using an object reference (if they aren’t private)

**Class Variables (Static variables)**

These are declared within a class with the ‘static’ keyword, outside any method, constructor, or block. These variables are associated with the class itself rather than any instance. Just like instance variables, they can also be accessed from outside the class using the class name (if they are not private)

1. **Why is the initialization of variables required?**

Initialization of variables in Java is required to ensure that they have a well-defined value before use. Without using them, the program is prone to errors.

**4. Differentiate between static, instance, and local variables**

Static variables are declared within a class with the ‘static’ keyword, these variables are accessible throughout the class and through the class itself. They exist for the duration of the program, belong to the class itself, and are shared among all instances.

Instance variables are declared inside a class, outside any method, these variables are accessible throughout the class but exist as long as the object exists. They belong to an instance of the class.

Local variables are variables declared inside a method, constructor, or block where declared. They are only accessible within the method, constructor, or block that is declared. They exist only during the execution of the method, constructor, or block.

**5.**  **Differentiate between widening and narrowing casting in Java**

**i) Widening Casting (Automatic Type Conversion):** It's defined as converting a smaller primitive data type automatically. Examples include; byte-short-int-long–float-double. This method of conversion is safe because there is no loss of data.

ii) **Narrowing Casting (Explicit Type Conversion):** Converting a larger primitive type to a smaller primitive type manually. Examples include double-float-long-int-short-byte. This method of conversion can lead to data loss or precision loss because the larger type may contain more information than the smaller type can handle.

6. **The following is a table that shows data type, its size, default value and the range. Fill in the missing values.**

|  |  |  |  |
| --- | --- | --- | --- |
| **TYPE** | **SIZE (IN BYTES)** | **DEFAULT** | **RANGE** |
| Boolean | 1 bit | **false** | True, false |
| Char | 2 | **‘\u0000’** | **‘**\u0000’ to ‘\ffff’ |
| Byte | **1** | 0 | **-128 to 127** |
| Short | **2** | 0 | -215  to +215  -1 |
| Int | 4 | **0** | **-2.15\*109  to +2.15\*109** |
| Long | **8** | 0L | **-9.22\*1018  to +9.22\*1018** |
| Float | 4 | 00.0f | **-3.40\*1038  to +3.40\*1038** |
| Double | 8 | **0.0d** | -1.8E+308 to +1.8E+308 |

7. **Define class as used in OOP**

A class in Object-oriented programming is a blueprint or template for creating objects (instances of the class). It defines the structure and behavior that objects of the class will have. It encapsulates data and behaviors into a single unit to promote modularity and reusability in code.

8. **Explain the importance of classes in Java programming**

**a) Encapsulation:** this is the bundling of data(attributes) and methods(functions) that operate on the data into a single unit. This helps to organize and structure code by hiding sensitive data from the outside world and allowing controlled access through methods.

**b) Abstraction:** They create abstract data types that represent real-world entities or concepts in a simplified manner. By defining a class, developers can focus on essential characteristics of an object and hide irrelevant details.

**c) Code reusability:** This is to allow objects to be used multiple times across different parts of a program or in different programs altogether. Once a class is defined, it can be easily reused without the need to rewrite the code.

**d) Inheritance: In** Java, this happens when one class (subclass or derived class) can inherit the properties and behaviors of another class (superclass or base class). This promotes hierarchical relationships among classes and facilitates code reuse and extensibility.

**e) Polymorphism:** Classes enable polymorphism, which allows objects of different classes to be treated as objects of a common superclass. This feature is essential for achieving flexibility and extensibility in software design.

**f) Modularity: Classes** promote modularity by breaking down a complex system into smaller, manageable units. Each class represents a module that encapsulates a specific functionality or aspect of the system, making the structure more organized and easier to maintain.

**g) Data Integrity:** Encapsulating data within classes and providing controlled access through methods (getters and setters), classes help to ensure data integrity and prevent unauthorized access and modification of data.

**h) Software development practices:** Classes support best practices in software development, i.e. abstraction, modularity, and separation of concerns. They contribute to writing cleaner, neater and more maintainable, scalable code.