**Mathiuu Iguthu Maina**

**Reg No:23/01942**

**Java programming**

**Exercise 1 of 16**

**Instructions:**

All programs should be written, and linked to an online repository like GitHub.

<https://github.com/lordIguthu/MathiuuIguthu-23.01942-JavaAssignment>

A video to get you started with GitHub has been posted on Moodle.

After completing your assignment, post the link on the link on Moodle. An instructor will follow the posted link to access and grade your work.

Note that: Your program should always be well-commented. At the top of your source code file, you should write a short description of what your program does and add other comments to help in explaining your code.

All of your variables should be given a deceptive name. Avoid giving your variables names like a, b, I, x, y etc.

In case you copy your friend's work, you both get a Zero (0).

**Section 1:**

1. **Explain the differences between primitive and reference data types.**
2. Storage- Primitive is stored in the stack while reference is stored in the heap.
3. Value vs. Reference- Primitive contains the actual value while reference contains a reference to the value's memory address.
4. Memory Management- Primitive is managed directly by the language runtime while reference is managed through dynamic memory management (heap), often involving garbage collection.
5. Performance- Primitive is generally faster due to their simplicity and fixed size while reference can be slower due to additional overhead for reference handling and dynamic memory allocation.
6. **Define the scope of a variable (hint: local and global variable**

**Local Variable:**

* A local variable is declared within a specific block of code, such as a function, loop, or conditional statement.
* Its scope is limited to that block. It can only be accessed and used within the block where it's declared. Once the code execution exits that block, the local variable becomes inaccessible.
* Local variables help prevent naming conflicts with variables in other parts of your code and promote better code organization.

**Global Variable:**

* A global variable is declared outside of any function or block, typically at the beginning of your program.
* Its scope encompasses the entire program. You can access and modify a global variable from any part of your code.
* While convenient for sharing data across different parts of your program, overuse of global variables can lead to code that's harder to understand and maintain. They can introduce unintended side effects and make debugging more challenging.

1. **Why is initialization of variables required.**
2. **Predictable Behavior:** Uninitialized variables contain indeterminate values. This can lead to unexpected results and errors in your program's logic. By initializing variables, you ensure they start with a known value, making your code more predictable and reliable.
3. **Prevents Errors:** Certain operations, like mathematical calculations or using a variable in a conditional statement, might not work correctly if the variable hasn't been assigned a value. Initialization helps avoid these runtime errors that can crash your program.
4. **Improves Code Readability:** When you initialize a variable with a meaningful value, it becomes clear what kind of data it's meant to hold and how it's used in your code. This makes your code easier to understand for yourself and others who might read or maintain it later.
5. **Memory Management:** Uninitialized variables might still occupy memory space, even though the value is unknown. Proper initialization helps manage memory usage efficiently.
6. **Default Values:** Some languages provide default values for certain data types when you declare a variable without explicitly initializing it. While convenient, these defaults might not be what you expect, potentially causing issues. Explicit initialization ensures you have the intended value from the start.
7. **Differentiate between static, instance and local variables.**

**1.Scope**

* Local Variables- Limited to the block of code where they are declared
* Instance Variables- Accessible throughout the object's lifetime
* Static Variables- Shared by all instances of the class. There's only one copy of the static variable for the entire class

**2. Lifetime -**

* Local Variables- They exist only as long as the code execution is within that block. Once the code exits the block, the local variable is destroyed
* Instance Variables- They exist as long as the object (instance) exists. When the object is destroyed, its instance variables are also destroyed
* Static Variables- They exist throughout the program's execution, as long as the class is loaded in memory.

**3. Initialization -**

* Local Variables- Must be explicitly initialized before use to avoid errors
* Instance Variables- Can be explicitly initialized during object creation or have default values assigned
* Static Variables- Can be explicitly initialized during class definition or have default values assigned.

**4.Access –**

* Local Variables- Only accessible within the block where they are declared.
* Instance Variables- Accessed using the object reference followed by the variable name
* Static Variables- Accessed using the class name followed by the variable name

1. **Differentiate between widening and narrowing casting in java.**

Widening and narrowing casting are two ways to convert between data types in Java.

* **Conversion:** Widening converts a smaller data type to a larger data type while narrowing converts a larger data type to a smaller data type
* **Safety:** Widening is generally considered safe because there's no loss of data since the smaller value fits comfortably within the larger type while narrowing can be risky since if the larger value doesn't fit within the smaller type's range, data loss can occur which might lead to unexpected results or errors.
* **Java terminology:** Widening uses upcasting which refers to moving up the hierarchy of data types while narrowing uses downcasting which refers to moving down the hierarchy of data types
* **Performed implicitly:** Widening uses no explicit casting syntax as java performs the conversion automatically when necessary while narrowing requires casting syntax using parentheses and the target data type to tell Java you intend to convert the value

1. the following table shows data type, its size, default value and the range. Filling in the missing values.

|  |  |  |  |
| --- | --- | --- | --- |
| **TYPE** | **SIZE (IN BYTES)** | **DEFAULT** | **RANGE** |
| boolean | 1 bit | false | true, false |
| Char | 2 | u0000” | ‘\0000’ to ‘\ffff’ |
| Byte | 1 | 0 | -128 to 127 |
| Short | 2 | 0 | -215 to +215-1 |
| Int | 4 | 0 | -2^31 to 2^31-1 |
| Long | 8 | 0L | - 2^63 to 2^63-1 |
| Float | 4 | 00.0f | 1.4E-45 to 3.4E+38 |
| Double | 8 | 0.0 | -1.8E+308 to +1.8E+308 |

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

* A class is a blueprint or template that defines the properties and functionalities of objects. It acts like a cookie cutter that you can use to create multiple objects of the same kind.

1. **Explain the importance of classes in Java programming.**
2. **Object Creation:** Classes serve as blueprints for creating objects. They define the attributes and methods that objects of a particular type will possess. This allows you to create multiple objects with similar characteristics and behaviors, but potentially unique data values.
3. **Code Reusability:** By encapsulating functionalities within a class, you can reuse the code across different parts of your program.
4. **Code Organization:** Classes promote modularity by grouping related attributes and methods together. This makes your code more organized, easier to understand, and maintain.
5. **Data Encapsulation:** Classes can encapsulate data and methods within a single unit. This allows you to control access to the data using access modifiers preventing unintended modifications from other parts of your code and promoting data integrity.
6. **Inheritance:** Classes enable inheritance. You can create new classes that inherit attributes and methods from existing classes. This promotes code reusability and allows for specialization of objects
7. **Polymorphism:** Polymorphism allows objects of different classes to respond differently to the same method call. This makes code more flexible and adaptable.

Section 2:

1. Write a Java program that asks the user to enter their sur name and current age then print the number of characters of their sir name and even or odd depending on their age number.

Example of Expected result:

If sir name is Saruni and age is 29, output will be;

then the number of characters is 6.

Your current age is an odd number

<https://github.com/lordIguthu/MathiuuIguthu-23.01942-JavaAssignment/blob/main/SurNamelength.java>

1. Write Java program to ask student to enter the marks of the five units they did last semester, compute the average and display it on the screen. (Average should be given in two decimal places).

<https://github.com/lordIguthu/MathiuuIguthu-23.01942-JavaAssignment/blob/main/AverageMarks.java>

1. Write a program that will help kids learn divisibly test of numbers of integers. The program should check whether the given integer is divisible by integers in the range of 0-9. For example, if a number (955) is divisible by five, the program should print, the number is divisible by 5 because it ends with a 5, and 900 is divisible by 5 because it ends with a 0(zero).

<https://github.com/lordIguthu/MathiuuIguthu-23.01942-JavaAssignment/blob/main/DivisibilityTest.java>

1. Write a Java program to display all the multiples of 2, 3 and 7 within the range 71 to 150.

<https://github.com/lordIguthu/MathiuuIguthu-23.01942-JavaAssignment/blob/main/MultipleNumbers.java>

1. Create a calculator using java to help user perform the basic operations (+, -, \* and /).
   1. User should be asked to enter a number, then an operation, the program computes the operation and display the output to the computer screen.

<https://github.com/lordIguthu/MathiuuIguthu-23.01942-JavaAssignment/blob/main/SimpleCalculator.java>