# Java programming

**Exercise 1 of 16**

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

| | Primitive data type | Reference data type |
|---|---|---|
| Definition | They are not objects and hold simple values. | These are objects and arrays that store references (addresses) to the actual data. |
| Types | byte, short, int, long, float, double, char, boolean. | Classes, Interfaces, Arrays, Enums. |
| Size | Fixed size depending on the type | The size of the reference is typically fixed but the actual object size can vary. |
| Default Values | Each primitive type has a default value | The default value for reference types is `null`. |
| Storage | Stored directly in the memory location associated with the variable. | stores a reference (memory address) to the actual data in the heap memory. |
| | | |

2. Define the scope of a variable (hint: local and global variable)

- The scope of a variable determines the part of the program where the variable can be accessed and modified.

| | Local variable | Global variable |
|---|---|---|
| Definition | are declared within a method, constructor, or block. | are declared outside any method, constructor, or block. |
| Scope | are accessible only within the method, constructor, or block where they are declared. | are accessible throughout the class via object reference. |
| Lifetime | They exist only during the execution of that method, constructor, or block. | They exist only during the execution of that method, constructor, or block. |

3. Why is initialization of variables required.

    - **Avoid Garbage Values**: Uninitialized variables can hold any arbitrary value leading to unpredictable behavior or errors.
    - **Logical Errors**: Ensures that the variable holds a valid, expected value before it is used.
    - **Code Clarity and Maintenance**: Makes the code easier to understand and maintain, as the initial values of variables are known.

4. Differentiate between static, instance and local variables.

    **Static Variables**

    - Associated with the class itself, not specific instances (objects).
    - Shared across all objects of the same class.
    - Declared using the static keyword.

    **Instance Variables**

    - Associated with individual instances (objects) of a class.
    - Each object has its own copy of instance variables.
    - Declared within the class but outside any methods or constructors.

    **Local Variables**

    - Declared within a method, constructor, or block.
    - Limited to the scope where they are defined.
    - Used for temporary storage within a specific context3.

5. Differentiate between widening and narrowing casting in java.

    **Widening Casting**

    - Automatic type conversion from a smaller type to a larger type.
    - No explicit syntax needed.
    - Example: int to long, float to double.

    **Narrowing Casting**

    - Manual type conversion from a larger type to a smaller type.
    - Requires explicit casting (float)).
    - Example: double to float, long to int4.

6. 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 | 8 | 0 | -128 to 127 |
| Short | 16 | 0 | $-2^{15}$ to $+2^{15}$-1 |
| Int | 4 | 0 | -2^31 to 2^31-1 |
| Long | 64 | 0L | -2^63 to 2^63-1 |
| Float | 4 | 00.0f | ~1.4E-45 to ~3.4E+38 |
| Double | 8 | 0.0d | -1.8E+308 to +1.8E+308 |

7. Define class as used in OOPz

   - A **class** in object-oriented programming (OOP) serves as a blueprint or template for creating objects. It encapsulates both data and behavior related to a specific type of entity.

   Key points about classes:
   - **Blueprint**: A class defines the structure and characteristics of objects that belong to that class.
   - **Properties**: It specifies the attributes (variables) that an object of the class will have.
   - **Methods**: It defines the behaviors (functions) that can be performed on those objects.
8. Explain the importance of classes in Java programming.

   - **Modularity**: Classes allow you to break down complex systems into smaller, manageable units.
   - **Reusability**: you can create multiple objects (instances) from it.
   - **Encapsulation**: Classes hide data and methods, ensuring that related functionality is grouped together and hides implementation details from other parts of the program.
   - **Inheritance**: Classes support inheritance, allowing you to create new classes based on existing ones hence promotes code sharing and hierarchy.
   - **Polymorphism**: Through classes, you can achieve writing code that works with objects of different classes but related by inheritance.
   - **Organization:** Classes provide a structured way to organize your code, making it easier to maintain and understand.

## 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

```
import java.util.Scanner;

public class SurnameAgeDetails {
    public static void main(String[] args) {
        // Create a Scanner object to read input
        Scanner scanner = new Scanner(System.in);
        // Prompt the user to enter their surname
        System.out.print("Enter your surname: ");
        String surname = scanner.nextLine();

        // Prompt the user to enter their current age
        System.out.print("Enter your current age: ");
        int age = scanner.nextInt();

        // Calculate the number of characters in the surname
        int surnameLength = surname.length();

        // Determine if the age is even or odd
        String ageType = (age % 2 == 0) ? "even" : "odd";

        // Display the results
        System.out.println("The number of characters in your surname is: " +
surnameLength);
        System.out.println("Your current age is an " + ageType + " number");

        // Close the scanner
        scanner.close();
    }
}
```

2. 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).

```
import java.util.Scanner;

public class AverageMarks {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double[] marks = new double[5]; // Array to store marks of five units

        // Prompt user to enter marks for five units
        for (int i = 0; i < 5; i++) {
            System.out.print("Enter the marks for unit " + (i + 1) + ": ");
            marks[i] = scanner.nextDouble();
        }

        // Calculate the average of the marks
        double total = 0;
        for (double mark : marks) {
            total += mark;
        }
        double average = total / marks.length;

        // Display the average with two decimal places
        System.out.printf("The average of the marks is: %.2f\n", average);

        scanner.close();
    }
}
```

3. 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).

```
import java.util.Scanner;

public class DivisibilityTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter an integer
        System.out.print("Enter an integer: ");
        int number = scanner.nextInt();

        // Check divisibility from 0 to 9
        for (int i = 0; i <= 9; i++) {
            if (i == 0) {
                System.out.println("Divisibility by 0 is undefined.");
            } else {
                if (number % i == 0) {
                    System.out.println(number + " is divisible by " + i);
                } else {
                    System.out.println(number + " is not divisible by " + i);
                }
            }
        }

        scanner.close();
    }
}
```

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

```
public class Multiples {
    public static void main(String[] args) {
        System.out.println("Multiples of 2, 3, and 7 between 71 and 150:");

        // Loop through the range 71 to 150
        for (int i = 71; i <= 150; i++) {
            if (i % 2 == 0 || i % 3 == 0 || i % 7 == 0) {
                System.out.println(i);
            }
        }
    }
}
```

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

```
import java.util.Scanner;

public class SimpleCalculator {
    public static void main(String[] args) {
        // Create a Scanner object to read user input
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter the first number
        System.out.print("Enter the first number: ");
        double firstNumber = scanner.nextDouble();

        // Prompt the user to enter an operator (+, -, *, /)
        System.out.print("Enter an operation (+, -, *, /): ");
        char operation = scanner.next().charAt(0);

        // Prompt the user to enter the second number
        System.out.print("Enter the second number: ");
        double secondNumber = scanner.nextDouble();

        // Variable to store the result of the operation
        double result;

        // Perform the calculation based on the operation
        switch (operation) {
            case '+':
                result = firstNumber + secondNumber;
                break;
            case '-':
                result = firstNumber - secondNumber;
                break;
            case '*':
                result = firstNumber * secondNumber;
                break;
            case '/':
                // Check if the second number is not zero to avoid division by zero
                if (secondNumber != 0) {
                    result = firstNumber / secondNumber;
                } else {
                    System.out.println("Error: Division by zero.");
                    scanner.close();
```

```java
                    return; // Exit the program if division by zero occurs
                }
                break;
            default:
                System.out.println("Invalid operation.");
                scanner.close();
                return; // Exit the program if an invalid operation is entered
        }

        // Display the result of the calculation
        System.out.println("The result of the operation is: " + result);

        // Close the scanner
        scanner.close();
    }
}
```