Java Assignment 1

22/02398

Titus Kitavi

Section 1

Q1. Explain the differences between primitive and reference data types.

1. Definition:
   - Primitive Data Types: are basic data types that are built-in to a programming language. They are predefined and represent simple values. Examples of primitive data types include integers, floating-point numbers, and characters.
   - Reference Data Types: are more complex data types that are defined by the programmer. They are created using classes or structures and contain references to memory locations where the actual data is stored. Examples of reference data types include arrays, strings, objects, and user-defined classes.
2. Storage and Memory:
   - Primitive Data Types: are stored directly in memory and have a fixed size.
   - Reference Data Types: are stored indirectly in memory.
3. Value vs. Reference Assignment:
   - Primitive Data Types: When a primitive variable is assigned to another variable, the value of the variable is copied. Each variable holds its own independent value, and modifying one variable does not affect the others.
   - Reference Data Types: When a reference variable is assigned to another variable, only the memory address (reference) is copied, not the actual data. Both variables then refer to the same object or data in memory.
4. Default Values:
   - Primitive Data Types: Each primitive data type has a default value assigned by the programming language. For example, the default value of an integer is 0.
   - Reference Data Types: Have a default value of null, which indicates that the reference does not currently point to any valid object or data.
5. Passing as Parameters:
   - Primitive Data Types: When passed as a parameter to a method or function, the value of the primitive variable is passed, and any modifications made within the method do not affect the original variable.
   - Reference Data Types: When passed as a parameter, the reference to the data is passed, allowing the method to access and modify the actual data.

Q2. Define the scope of a variable (hint: local and global variable)

The scope of a variable refers to the region or portion of a program where the variable is recognized, accessible, and can be used.

1. Local Variables:
   - Local variables are declared within a specific block of code, such as a function or a loop.
   - They have limited visibility and are accessible only within the block of code where they are declared.
   - Local variables are typically temporary and exist only during the execution of the block of code.
   - Once the block of code is exited, the local variable is destroyed, and its memory is released.

- Local variables can have the same name in different blocks of code since they are independent of each other.

2. Global Variables:
   - Global variables are declared outside of any specific block of code, typically at the top of a program.
   - They have global scope and can be accessed and modified from any part of the program, including functions, loops, and other blocks of code.
   - Global variables are visible to all functions and blocks within the program, making them accessible throughout the entire program's execution.
   - Global variables are typically used for values that need to be shared and accessed by multiple parts of the program.
   - Modifying the value of a global variable in one part of the program will affect its value in all other parts of the program.

Q3. Why is initialization of variables required.

Initialization of variables is required in programming to assign an initial value to a variable before it is used in computations or operations.

why it is important to initialize.

1. Avoiding Garbage Values: When a variable is declared without initialization, it can contain a random or undefined value.
2. Preventing Bugs and Errors: Initializing variables helps prevent bugs and errors caused by using uninitialized variables.
3. Providing Default Values: Initialization allows you to provide default values for variables.
4. Ensuring Consistent Behavior: By initializing variables, you establish a consistent starting point for their values.
5. Enhancing Readability and Maintainability: Initializing variables explicitly makes your code more readable and understandable.

Q4. Differentiate between static, instance and local variables.

Static variables, also known as class variables, are associated with a class rather than with instances or objects of the class.

Instance variables, also known as member variables or object variables, are unique to each instance or object of a class.

Local variables are declared within a method, constructor, or block of code and have limited scope.

Q5. Differentiate between widening and narrowing casting in java.

Widening Casting :Widening casting occurs when you convert a value of a smaller data type to a value of a larger data type.

Narrowing Casting :Narrowing casting occurs when you convert a value of a larger data type to a value of a smaller data type.

Q6. 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	`0\'	'\0000' to '\ffff'
Byte	1	0	-128 to +127
Short	2	0	-215 to +215-1
Int	4	0	-2,147,483,648 to +2,147,483,647
Long	8	0L	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
Float	4	00.0f	-3.4+38 to +3.4+38

Double	8	0.0d	-1.8E+308 to +1.8E+308

Q7. Define package as used in java programming.

A package is essentially a directory that contains the Java source files (.java) and compiled bytecode files (.class) associated with the classes within that package. The package declaration appears at the beginning of a Java source file and specifies the package to which the file belongs.

Q8. Explain the importance of using Java packages.

1. Namespace Management: Packages help prevent naming conflicts by providing a namespace for classes. Since each package has a unique name, you can avoid conflicts when using classes with the same name from different packages.
2. Access Control: Packages provide access control through the use of access modifiers like public, protected, and private. Classes and resources within a package have default package-level access, meaning they are accessible to other classes within the same package.
3. Code Reusability: Packages encourage code reusability by providing a mechanism for creating libraries and modules. By packaging related classes and resources together, you can create reusable components that can be easily shared across different projects or within a team.
4. Organization and Modularity: Packages provide a way to organize and structure code logically. They allow you to group related classes, interfaces, and resources together, making it easier to locate and manage code.

SECTION 2.

Q1. 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

```
/*The program uses the Scanner class from the java.util package to read user input.
*The user is prompted to enter their surname using System.out.print and scanner.nextLine().
*The user is prompted to enter their age using System.out.print and scanner.nextInt().
*The length() method is used to determine the number of characters in the surname.
*The result is printed using System.out.println.
*The age is checked for evenness or oddness using the modulus operator %. If the remainder is 0, it is even; otherwise, it is odd.
*The result is printed using System.out.println.
*/
import java.util.Scanner;

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

        // Asking for surname
        System.out.print("Enter your surname: ");
        String surname = scanner.nextLine();

        // Asking for age
        System.out.print("Enter your current age: ");
```

```
        int age = scanner.nextInt();

        // Printing the number of characters in the surname
        int surnameLength = surname.length();
        System.out.println("The number of characters in your surname is: " + surnameLength);

        // Checking if the age is even or odd
        String evenOdd = (age % 2 == 0) ? "even" : "odd";
        System.out.println("Your current age is an " + evenOdd + " number");

        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).

```
/*The program uses the Scanner class from the java.util package to read user input.
*The user is prompted to enter the marks for each of the five units using a for loop.
*The nextDouble() method is used to read the double values entered by the user.
*The sum of the marks is calculated by adding each input value to the sum variable.
*After the loop, the average is calculated by dividing the sum by 5 (the number of units).
*The printf() method is used to display the average with two decimal places using the format specifier %.2f.
*The scanner.close() method is called to close the scanner and free up system resources.
*/
import java.util.Scanner;

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

        // Asking for marks of five units
        System.out.println("Enter the marks of five units:");

        double sum = 0;

        for (int i = 1; i <= 5; i++) {
            System.out.print("Unit " + i + ": ");
            double marks = scanner.nextDouble();
            sum += marks;
        }

        // Calculating average
        double average = sum / 5;

        // Displaying average with two decimal places
        System.out.printf("The average 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).

```
/*The program uses the Scanner class from the java.util package to read user input.
*The user is prompted to enter an integer using System.out.print and scanner.nextInt().
*A for loop is used to iterate through integers in the range of 0-9.
*The % operator is used to check whether the number is divisible by the current integer in the loop. If the remainder is 0, it is d
*If the number is divisible by an integer, a message is printed using System.out.println.
*The scanner.close() method is called to close the scanner and free up system resources
*/
import java.util.Scanner;

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

```
        // Asking for the integer to be tested
        System.out.print("Enter an integer: ");
        int number = scanner.nextInt();

        // Checking divisibility by integers in the range of 0-9
        for (int i = 0; i <= 9; i++) {
            if (number % i == 0) {
                System.out.println("The number is 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.

```
/*The program defines the starting range as 71 and the ending range as 150.
*A for loop is used to iterate through the numbers within the given range.
*The % operator is used to check whether the current number is divisible by 2, 3, or 7. If the remainder is 0, it is a multiple of
*If the number is a multiple of 2, 3, or 7, it is printed using System.out.println.
*The loop continues until the end of the range is reached.
*The program displays all the multiples of 2, 3, and 7 within the range 71 to 150
*/
public class MultiplesInRange {
    public static void main(String[] args) {
        int startRange = 71;
        int endRange = 150;

        System.out.println("Multiples of 2, 3, and 7 within the range 71 to 150:");

        for (int i = startRange; i <= endRange; 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.

```
/*The program uses the Scanner class from the java.util package to read user input.
*The user is prompted to enter the first number, the operation (+, -, *, or /), and the second number.
*The nextDouble() method is used to read the double values entered by the user.
*A switch statement is used to perform the selected operation based on the user's input.
*The result of the operation is stored in the result variable.
*If the user attempts to divide by zero, an error message is displayed, and the program exits.
*If an invalid operation is entered, an error message is displayed, and the program exits.
*The result is displayed using System.out.println.
*The scanner.close() method is called to close the scanner and free up system resources.
*/
import java.util.Scanner;

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

        // Asking for the first number
        System.out.print("Enter the first number: ");
        double num1 = scanner.nextDouble();

        // Asking for the operation
        System.out.print("Enter the operation (+, -, *, /): ");
        String operation = scanner.next();

        // Asking for the second number
        System.out.print("Enter the second number: ");
```

```
        double num2 = scanner.nextDouble();

        double result = 0;

        // Performing the operation and calculating the result
        switch (operation) {
            case "+":
                result = num1 + num2;
                break;
            case "-":
                result = num1 - num2;
                break;
            case "*":
                result = num1 * num2;
                break;
            case "/":
                if (num2 != 0) {
                    result = num1 / num2;
                } else {
                    System.out.println("Error: Division by zero is not allowed.");
                    System.exit(0);
                }
                break;
            default:
                System.out.println("Error: Invalid operation entered.");
                System.exit(0);
        }

        // Displaying the result
        System.out.println("Result: " + result);

        scanner.close();
    }
}
```