Q1.

Primitive data type are blocks of data in programming they include integer, numbers, characters and boolean values . And these are considered primitive because they can not be broken down

Reference data types, this are complex data types that can contain other data types they include arrays, lists.

The main difference is that primitive are stored in single value in the memory while reference data type are stored as a collection of values in memory.

Q2.

The scope of a variable referes to the region of a program where that variables can be accessed and used .

Local variable: This is defined within a methode or function and its scope is limited to specific blocks. It cannot be accessed outside of that block.

Global variable: This is defined outside of any method, function or block. This can be accessed from any part of the program.

Q3.

Initialization of variables is required to ensure that it has a value assigned to it when it is created, privet null pointer exceptions or other retime errors, and make the code more readable and understandable.

Q4.

Static variables are shared by all instance of a class, instance variable are associated with a specific instance of a class, and local variables have a limited scope within a specific block or method.

Q5.

Widening casting involves automatically converting a value of a smaller data type to a larger data type, while narrowing casting involves explicitly converting a value of a large data type to a smaller datatype. Widening casting is typically used when assigning a value to a variable of a larger data type, while narrowing casting is typically used when assigning a value to a variable of a smaller data type.

Q6.

TYPE	SIZE (IN BYTES)	DEFAULT	RANGE
boolean	1 bit	False	true, false
Char	2 bytes		'\0000' to '\ffff'
Byte	1 byte	0	-128 to 127
Short	2 bytes	0	-2^{15} to $+2^{15}-1$
Int	4 bytes	0	-2^31 to 2^31-1

Long	8 bytes	0L	- 2^31 to 2^63-1
Float	4 bytes	00.0f	-3.4e38 to 1.2e28
Double	8 bytes	0.0d	-1.8E+308 to +1.8E+308

Q7.

Object-oriented programming is a blueprint or templet for creating object. And this allow for code reusability, modularity and encapsulation.

Q8.

Code reusability: This allow one to define a blueprint for creating objects with similar behavior

Modularity: Classes promote modularity by allowing you to organize your code into separate files, each containing a single class.

Polymorphism: Class support this which means that objects of different classes can be treated as object of different classes can treated as object of their parent class.

SECTION2

Q1.

```
import java.util.Scanner;


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


      // Prompt the user to enter their surname
      System.out.print("Please enter your surname: ");
      String surname = scanner.nextLine();
```

```
        // Prompt the user to enter their age
        System.out.print("Please 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 ageParity = (age % 2 == 0) ? "even" : "odd";

        // Print the results
        System.out.println("The number of characters in your surname is " + surnameLength + ".");
        System.out.println("Your current age is an " + ageParity + " number.");
    }
}
```

Q2.

```
import java.util.Scanner;

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

        // Array to store the marks of five units
        double[] marks = new double[5];
        double sum = 0;

        // Prompt the student to enter marks for five units
        for (int i = 0; i < 5; i++) {
```

```
            System.out.print("Please enter the marks for unit " + (i + 1) + ": ");
            marks[i] = scanner.nextDouble();
            sum += marks[i]; // Add the mark to the sum
        }

        // Calculate the average
        double average = sum / 5;

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

Q3.

```
import java.util.Scanner;

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

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

        // Loop through the range of 1 to 9
        for (int i = 1; i <= 9; i++) {
            if (isDivisible(number, i)) {
                System.out.println("The number " + number + " is divisible by " + i + ".");
                // Additional explanation for divisibility by 5
```

```
            if (i == 5) {
                if (number % 10 == 0) {
                    System.out.println("This is because it ends with a 0.");
                } else if (number % 10 == 5) {
                    System.out.println("This is because it ends with a 5.");
                }
            }
        } else {
            System.out.println("The number " + number + " is not divisible by " + i + ".");
        }
    }
  }

  // Method to check divisibility
  public static boolean isDivisible(int number, int divisor) {
    return number % divisor == 0;
  }
}
```

Q4.

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

```
}
```

Q5.

```
import java.util.Scanner;

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

        // Prompt the user to enter the first number
        System.out.print("Enter the first number: ");
        double num1 = scanner.nextDouble();

        // Prompt the user to enter an operation
        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 num2 = scanner.nextDouble();

        // Variable to store the result
        double result = 0;

        // Perform the operation based on user input
        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.");
                    return; // Exit the program
                }
                break;
            default:
                System.out.println("Error: Invalid operation.");
                return; // Exit the program
        }

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