1. Differences between Primitive and Reference Data Types

In Java, primitive data types are the basic building blocks. They store actual values directly in memory and include types like int, char, and boolean. For instance, an int can hold whole numbers from -2 billion to 2 billion.

Reference data types, on the other hand, don't store the actual data. Instead, they store references (or addresses) to the objects in memory. This includes things like arrays and classes. When I create an object, what I'm really doing is creating a reference to that object, not the object itself. It's essential to know this difference because it affects memory usage and how data is accessed.

2. Scope of a Variable

The scope of a variable determines where it can be accessed in the code. **Local variables** are defined within a method or block. They can only be accessed within that method. For example, if I declare a variable inside a loop, I can't use it outside of that loop.

Global variables, or **instance variables**, are defined at the class level. They can be accessed by any method within the class, making them useful for storing data that needs to be shared across methods.

3. Why Initialization of Variables is Required

Initializing variables is crucial because if I use a variable without setting a value, it can lead to unpredictable behavior. Java initializes some types to default values, but if I forget to initialize, I might encounter errors or unexpected results. So, always initializing my variables helps ensure that they have a known starting value before I use them.

4. Differences between Static, Instance, and Local Variables

- **Static Variables**: These belong to the class rather than any instance. All instances share the same static variable, which can be handy for values that should be common across instances, like a counter for how many objects of the class have been created.
- **Instance Variables**: Each object has its own copy of instance variables. This allows objects to maintain their own state, which is essential for object-oriented programming.
- **Local Variables**: These are temporary and are created when a method is called. They only exist while the method is executing, making them great for temporary calculations.

5. Widening vs. Narrowing Casting in Java

Widening casting is when I convert a smaller primitive type to a larger one, like changing an int to a float. This is safe and happens automatically.

Narrowing casting is the opposite. If I try to convert a larger type, like a double, to a byte, I need to do it explicitly, and there's a risk of losing data. I use parentheses to indicate that I know what I'm doing.

6. Data Types Table

TYPE	SIZE (IN BYTES)	DEFAULT	RANGE
boolean	1 bit	false	true, false
char	2	'\u0000'	'\u0000' to '\uffff'
byte	1	0	-128 to 127
short	2	0	-32,768 to +32,767
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	0.0f	±3.40282347E+38 (6-7 significant decimal digits)
double	8	0.0d	±1.79769313486231570E+308 (15 significant decimal digits)

7. Define Class in OOP

A class in OOP is like a blueprint for creating objects. It defines the properties (attributes) and methods (behaviors) that the objects created from the class will have. For example, if I have a class Car, it might have attributes like color and model, and methods like drive() and stop().

8. Importance of Classes in Java Programming

Classes are super important in Java because they help organize code and promote reusability. They allow us to create complex types and encapsulate data and behavior together, making our programs easier to manage and understand. By grouping related variables and methods, we can model real-world entities more naturally.

Section 2: Java Programs

1. User Input for Surname and Age

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


      System.out.print("Enter your surname: ");
      String surname = scanner.nextLine();
      System.out.print("Enter your current age: ");
```

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

        int surnameLength = surname.length();
        System.out.println("The number of characters is " + surnameLength + ".");
        System.out.println("Your current age is an " + (age % 2 == 0 ? "even" : "odd") + " number.");
    }
}
```

2. Average Marks Calculation

```
public class AverageMarks {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double totalMarks = 0;

        for (int i = 1; i <= 5; i++) {
            System.out.print("Enter marks for subject " + i + ": ");
            totalMarks += scanner.nextDouble();
        }

        double average = totalMarks / 5;
        System.out.printf("The average marks are: %.2f%n", average);
    }
}
```

3. Divisibility Test

```
public class DivisibilityTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter an integer: ");
```

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

        for (int i = 1; i <= 9; i++) {
            if (number % i == 0) {
                System.out.println(number + " is divisible by " + i);
            }
        }
    }
}
```

4. Multiples of 2, 3, and 7 in Range

```
public class MultiplesInRange {
    public static void main(String[] args) {
        System.out.println("Multiples of 2, 3, and 7 from 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. Basic Calculator

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

        System.out.print("Enter first number: ");
        double num1 = scanner.nextDouble();
        System.out.print("Enter an operation (+, -, *, /): ");
```

```
    char operation = scanner.next().charAt(0);
    System.out.print("Enter second number: ");
    double num2 = scanner.nextDouble();

    double 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");
                return;
            }
            break;
        default:
            System.out.println("Invalid operation");
            return;
    }
    System.out.println("The result is: " + result);
}
```

}