sandra

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

Primitive Data Types: Primitive data types, also known as basic data types, are built-in data types provided by the programming language. They are predefined and have a fixed size in memory. Examples of primitive data types include integers, floating-point numbers, characters. Primitive data types are stored directly in memory and are typically represented as a sequence of bits or bytes while **Reference Data Types**: Reference data types, also known as object types, are data types that refer to objects . They are typically created using class definitions or predefined reference types provided by the programming language. Reference data types do not store the actual data directly, but instead, store a reference or memory address pointing to the location where the data is stored in memory

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

The scope of a variable refers to the region of a program where the variable is accessible or visible. It determines where and how a variable can be referenced and manipulated within the code.

3. Why is initialization of variables required

Memory allocation: When a variable is declared, memory is allocated to store its value. Initialization ensures that the allocated memory is set to a known and defined value. If the variable is not initialized, it may contain garbage or random data that was previously stored in that memory location

Predictable behavior: Initializing variables helps in ensuring predictable behavior and consistent results in a program. By explicitly assigning an initial value, you can avoid potential bugs or unintended consequences that may arise from using uninitialized variables.

Avoiding undefined behavior: In some programming languages, using uninitialized variables can lead to undefined behavior. This means that the outcome of an operation or the behavior of the program is not defined or predictable.

Default values: Initialization allows you to set default values for variables. Default values can be useful when you want to provide a fallback or initial value that represents a specific state or condition

Readability and maintainability: Initializing variables can improve the readability and maintainability of your code. It makes the intention of the variable clear to other developers who may read or modify the code.

4. Differentiate between static, instance and local variables.

The local variable can be accessed by using the object of the Demo class, whereas the static variable can be accessed using the name of the class. Instance variable a variable which is declared in a class but outside of constructors.

5. Differentiate between widening and narrowing casting in java.

- widening casting implicit casting while narrowing casting explicit casting.

In widening casting,

- No Data Loss: Widening casting is considered safe because it involves converting to a larger data type, which can accommodate a wider range of values without loss of information.
- No Explicit Casting: The Java compiler performs Widening casting automatically, and no explicit casting syntax is required.

In narrow casting,

Potential Data Loss: Narrowing casting is potentially unsafe because it involves converting to a smaller data type, which may result in loss of information or precision.

Explicit Casting: Narrowing casting requires explicit casting syntax by placing the target data type in parentheses before the value being cast.

6. 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	1	0	-128 to 127
Short	2	0	-2^{15} to $+2^{15}-1$
Int	4	0	-2147,483,648 to 2147,483,647
Long	8	0L	- 9223,372,036,854,775,808 to 9223,372,036,854,775,807
Float	4	00.0f	+- 3.403E+38F
Double	8	0.0	-1.8E+308 to +1.8E+308

7. Define package as used in java programming

A container that groups related types Java classes, interfaces, enumerations, and annotations

8. Explain the importance of using Java packages

Code Reusability: Packages facilitate code reusability by allowing you to create libraries and modules

Code Readability and Maintenance: Packages contribute to code readability by providing a hierarchical structure that reflects the logical relationships between classes. Well-organized packages make it easier for developers to navigate and comprehend the codebase.

Organization and Modularity: Packages provide a structured way to organize classes. They help in creating a modular and well-organized codebase, making it easier to locate and manage related classes.

Documentation and API Design: Packages allow for better documentation by providing a natural grouping for related classes and functionality.

Collaboration and Team Development: Packages enhance collaboration among developers working on the same project. By providing a clear organization structure, packages make it easier for team members to understand the codebase, locate specific classes, and work on different parts of the project easily.

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

```
/*In this program, we use the Scanner class to read input from the user. First, we ask the user to enter their surname and store it in the

Next, we calculate the number of characters in the surname using the length() method of the String class and store it in the surnameLength

To determine whether the age is even or odd, we use the modulus operator % to check if the remainder of dividing age by 2 is 0. If it is, t

Finally, we print the number of characters in the surname and whether the age is even or odd using System.out.println() statements.
*/

import java.util.Scanner;

public class SurnameAndAge {

public static void main(String[] args) {

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 age

System.out.print("Enter your current age: ");

int age = scanner.nextInt();

// Calculate the number of characters in the surname

int surnameLength = surname.length();

// Print the number of characters in the surname

System.out.println("The number of characters in your surname is: " + surnameLength);

// Check if the age is even or odd and print the result

if (age % 2 == 0) {

System.out.println("Your current age is an even number");

} else {

System.out.println("Your current age is an odd number");
```

```
}

}

}
```

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).

```
/*In this program, we use the Scanner class to read input from the user. We prompt the user to enter the marks for each unit in a loop that

After the loop, we calculate the average by dividing the totalMarks by 5 (the number of units). We store the result in the average variable

Finally, we use System.out.printf() to display the average marks on the screen, formatted to two decimal places using the %.2f format speci
*/
import java.util.Scanner;

public class AverageMarks {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt the student to enter marks for five units

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

double totalMarks = 0;

for (int i = 1; i <= 5; i++) {

System.out.print("Unit " + i + ": ");

double marks = scanner.nextDouble();

totalMarks += marks;

}

// Compute the average

double average = totalMarks / 5;

// Display the average with two decimal places

System.out.printf("The average marks is: %.2f%n", average);

}

}
```

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).

```
/*In this program, we use the Scanner class to read an integer input from the user. The program checks whether the given number is divisibl

First, we initialize a boolean variable divisible to keep track of whether the number is divisible by any of the integers.

We check for divisibility by 2 by using the modulus operator %. If the remainder of dividing the number by 2 is 0, it means the number is e

Next, we check for divisibility by 3. We use the modulus operator again to check if the remainder of dividing the number by 3 is 0. Additio

For divisibility by 5, we check if the number is divisible by 5 using the modulus operator and if it ends with either 0 or 5. If both condi

Finally, we check for divisibility by 9 by checking if the remainder of dividing the number by 9 is 0. Similar to divisibility by 3, we exp

If none of the conditions for divisibility are met, we print a message stating that the number is not divisible by 2, 3, 5, or 9.
```

```
*/
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 by integers 0-9

for (int divisor = 0; divisor <= 9; divisor++) {

if (isDivisible(number, divisor)) {

System.out.println("The number is divisible by " + divisor);

}

}

}

// Check if a number is divisible by a given divisor

public static boolean isDivisible(int number, int divisor) {

if (divisor == 0) {

// Avoid division by zero

return number == 0;

} else if (divisor == 2) {

// Check divisibility by 2 (even number)

return number % 2 == 0;

} else if (divisor == 5) {

// Check divisibility by 5 (ends with 0 or 5)

return number % 10 == 0 || number % 10 == 5;

} else {

// Check divisibility by other divisors

return number % divisor == 0;

}

}

}
```

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

```
/*In this program, we use a for loop to iterate through the numbers from 71 to 150. For each number, we check if it is divisible by 2, 3, o

If the condition is true, we print the number using System.out.println().

Note that we use the logical OR operator || to combine the conditions for divisibility by 2, 3, and 7. This means that if the number satisf

When you run this program, it will display all the multiples of 2, 3, and 7 within the specified range of 71 to 150.
*/
public class Multiples {

public static void main(String[] args) {

int start = 71;
```

```
int end = 150;

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

for (int i = start; i <= end; 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 /).

User should be asked to enter a number, then an operation, the program computes the operation and display the output to the computer screen.

```
/*In this program, we use the Scanner class to read input from the user. First, we prompt the user to enter the first number and store it i

Then, we ask the user to enter the desired operation using one of the four accepted operators: +, -, *, or /. The operator is stored in the

Next, we prompt the user to enter the second number and store it in the num2 variable.

Using a switch statement, we perform the appropriate operation based on the operator entered by the user. The result is stored in the resul

Finally, we display the result on the screen using System.out.println().
*/
public class Multiples {

public static void main(String[] args) {

int start = 71;

int end = 150;

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

for (int i = start; i <= end; i++) {

if (i % 2 == 0 || i % 3 == 0 || i % 7 == 0) {

System.out.println(i);

}

}

}

}
```