Wasim akram

Id 11758

Subject oops

Date 30/ 09/ 2020

Jj

Q1 How many variables are being supported by java justify your answer with

A variable is assigned with a data type.

Variable is a name of memory location. There are three types of variables in java: local, instance and static

Local Variables: A variable defined within a block or method or constructor is called local variable.

- These variable are created when the block in entered or the function is called and destroyed after exiting from the block or when the call returns from the function.
- The scope of these variables exists only within the block in which the variable is declared. i.e. we can access these variable only within that block.
- Initilisation of Local Variable is Mandatory.

Sample Program 1:

```
public class StudentDetails {

        public void StudentAge()
{
        // local variable age
        int age = 0;
        age = age + 5;
        System.out.println("Student age is : " + age);
}
```

```
        public static void main(String args[])
        {
                StudentDetails obj = new StudentDetails();
                obj.StudentAge();
        }
}
```

Output:

```
Student age is : 5
```

Instance Variables: Instance variables are non-static variables and are declared in a class outside any method, constructor or block.

- As instance variables are declared in a class, these variables are created when an object of the class is created and destroyed when the object is destroyed.
- Unlike local variables, we may use access specifiers for instance variables. If we do not specify any access specifier then the default access specifier will be used.
- Initilisation of Instance Variable is not Mandatory. Its default value is 0
- Instance Variable can be accessed only by creating objects.

Sample Program:

```
import java.io.*;
class Marks {
    // These variables are instance variables.
    // These variables are in a class
    // and are not inside any function
    int engMarks;
    int mathsMarks;
    int phyMarks;
}

class MarksDemo {
    public static void main(String args[])
    {
        // first object
        Marks obj1 = new Marks();
        obj1.engMarks = 50;
        obj1.mathsMarks = 80;
```

```
obj1.phyMarks = 90;

// second object
Marks obj2 = new Marks();
obj2.engMarks = 80;
obj2.mathsMarks = 60;
obj2.phyMarks = 85;

// displaying marks for first object
System.out.println("Marks for first object:");
System.out.println(obj1.engMarks);
System.out.println(obj1.mathsMarks);
System.out.println(obj1.phyMarks);

// displaying marks for second object
System.out.println("Marks for second object:");
        System.out.println(obj2.engMarks);
        System.out.println(obj2.mathsMarks);
        System.out.println(obj2.phyMarks);
    }
}
```

Static Variables: Static variables are also known as Class variables.

- These variables are declared similarly as instance variables, the difference is that static variables are declared using the static keyword within a class outside any method constructor or block.
- Unlike instance variables, we can only have one copy of a static variable per class irrespective of how many objects we create.
- Static variables are created at the start of program execution and destroyed automatically when execution ends.
- Initilisation of Static Variable is not Mandatory. Its default value is 0
- If we access the static variable like Instance variable (through an object), the compiler will show the warning message and it won't halt the program. The compiler will replace the object name to class name automatically.
- If we access the static variable without the class name, Compiler will automatically append the class name.

To access static variables, we need not create an object of that class, we can simply access the variable as

```
class_name.variable_name;
```

Sample Program:

```
import java.io.*;
class Emp {

    // static variable salary
    public static double salary;
    public static String name = "Harsh";
```

```
}

public class EmpDemo {
    public static void main(String args[])
    {

        // accessing static variable without object
        Emp.salary = 1000;
        System.out.println(Emp.name + "'s average salary:"
                           + Emp.salary);
    }
}
```

:

Q2. Why “If” is used in java justify your answer with the help java coded example and explain in detail?

The **Java if statement** is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

Syntax:

```
if(condition)

{

   // Statements to execute if

   // condition is true

}
```

Working of if statement

- Control falls into the if block.
- The flow jumps to Condition.
- Condition is tested.
  - If Condition yields true, goto Step 4.
  - If Condition yields false, goto Step 5.
- The if-block or the body inside the if is executed.
- Flow steps out of the if block.
-

// Java program to illustrate If statement

class If {

```
public static void main(String args[])
{
        int i = 10;

        if (i < 15)
                System.out.println("10 is less than 15");

        // This statement will be executed
        // as if considers one statement by default
        System.out.println("Outside if-block");
}
```

Output:

```
10 is less than 15

Outside if-block
```

Q3. Why “if else if” is used in java justify your answer with the help java coded example and explain in detail?

Java if..else..if Statement

In Java, we have an if...else...if ladder, that can be used to execute one block of code among multiple other blocks.

```
if (expression1) {
   // codes
}
else if(expression2) {
   // codes
}
else if (expression3) {
   // codes
```

```
}
.
.
else {
    // codes
}
```

Here, `if` statements are executed from the top towards the bottom. As soon as the test expression is `true`, codes inside the body of that the `if` statement is executed. Then, the control of the program jumps outside the `if-else-if` ladder.

If all test expressions are `false`, codes inside the body of `else` is executed.

Example 3: Java if..else..if Statement

```
class Ladder {
    public static void main(String[] args) {

        int number = 0;

        // checks if number is greater than 0
        if (number > 0) {
            System.out.println("The number is positive.");
        }

        // checks if number is less than 0
        else if (number < 0) {
            System.out.println("The number is negative.");
        }
        else {
            System.out.println("The number is 0.");
        }
    }
}
```

Output:

```
The number is 0.
```

In the above example, we are checking whether the `number` is positive, negative or zero. Here, we have two test expressions:

- `number > 0` - checks if the `number` is greater than `0`
- `number < 0` - checks if the `number` is less than `0`

Here, the value of `number` is `0`. So both the test expression evaluates to `false`. Hence the statement inside the body of `else` is executed.

Q4. What are loops, why they are used in java and how many types of loops are being supported by java explain in detail?

Loops in Java

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true. Loops are a fundamental concept in programming. In order to understand what loops arfor loop Imagine a program which is required to output a particular value of a variable 800 times. Now we all know that the code for writing output is System.out.println("Text"); But in order to print this 800 times we will need to write the same line 800 times in the code. That would take up a lot of effort which is particularly just copy pasting the same sentence 800 times. Let us say that you have managed to copy paste the entire thing easily. Now if there is a different program which requires you to print the first 800 natural numbers. The copy paste method would not work because you still would have to go to all these lines and fit a number. That's where loops come in to play. Loops make it very easy to group all the code that's needed to be repetitively processed and throw it under scope. The loop does the remaining job.

- There are three types of loops in Java. while loop
- do-while loop

Java While Loop

The Java *while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.

Syntax:

```
while(condition){
//code to be executed
}
```

```
public class WhileExample {
public static void main(String[] args) {
    int i=1;
    while(i<=10){
        System.out.println(i);
    i++;
    }
}
}
```

for loop
The **Java** for **loop** is a control flow statement that iterates a part of the programs multiple times. The **Java** while **loop** is a control flow statement that executes a part of the programs repeatedly on the basis of given boolean condition.

Example explained

Statement 1 sets a variable before the loop starts (int i = 0).

Statement 2 defines the condition for the loop to run (i must be less than 5). If the condition is true, the loop will start over again, if it is false, the loop will end.

Statement 3 increases a value (i++) each time the code block in the loop has been executed.

```
for (int i = 0; i < 5; i++) {
```

```
  System.out.println(i);

}
```

Output

```
0
1
2
3
4
```

do-while loop

The Java *do-while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.

The Java *do-while loop* is executed at least once because condition is checked after loop body.

The example below uses a `do/while` loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

Example

```
int i = 0;
do {

  System.out.println(i);

  i++;

}

while (i < 5);
```

out put

```
0
1
2
3
4
```

Q5. Write 3's table in decremented form in java which takes input from user write java coded program and explain in detail

This is a Java Program to Print Multiplication Table for any Number.

Enter any integer number as input of which you want multiplication table. After that we use for loop from one to ten to generate multiplication of that number.