DAILY ASSESSMENT

Date:	**8-6-2020**	**Name:**	**Kavyashree m**
Course:	**Management and leadership**	**USN:**	**4al15ec036**
Topic:	**Modern leader training**	**Semester & Section:**	**8th A**
Github Repository:	**kavya**		

FORENOON SESSION DETAILS
Image of session

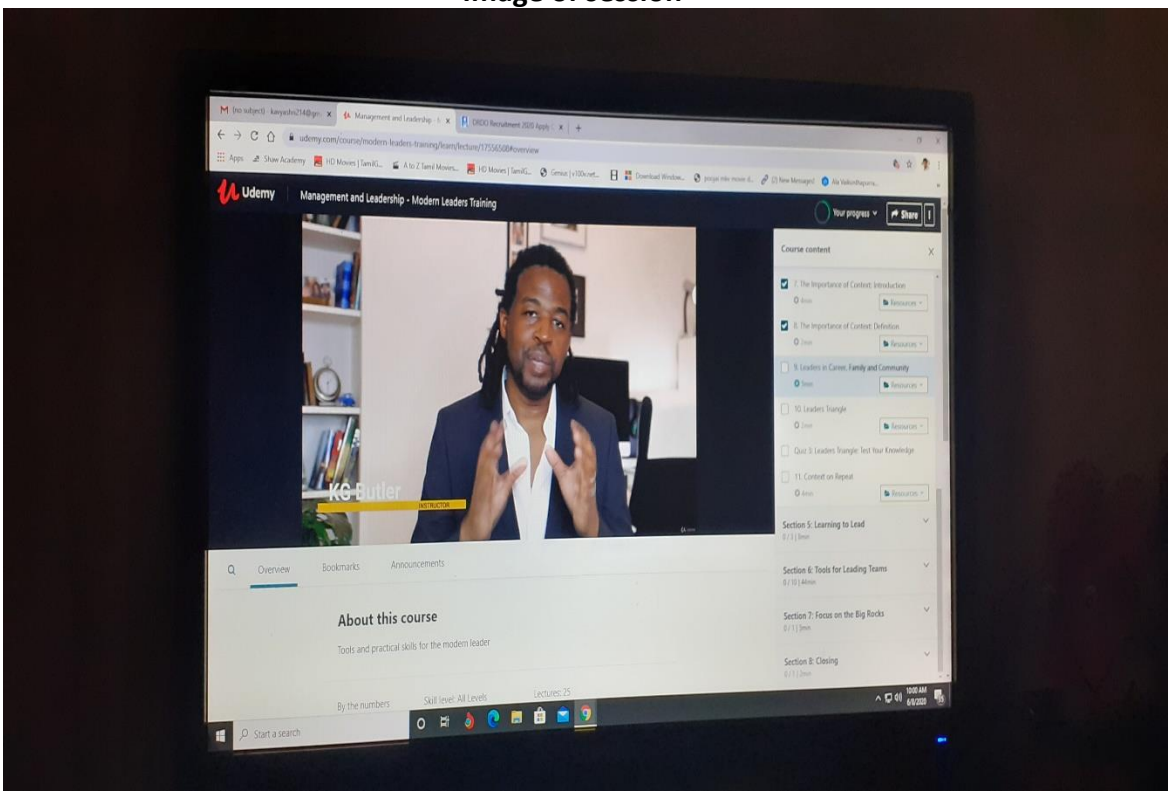


Fig 1: Modern leader training

Modern leader training

Leader

The person who leads or commands a group, organization, or country.

What is leadership?

A leader is someone who can see how things can be improved and who rallies people to move toward that better vision. Leaders can work toward making their vision a reality while putting people first. Just being able to motivate people isn't enough leaders need to be empathetic and connect with people to be successful.

Leader Vs Manager

A managerial culture emphasizes rationality and control. Whether his or her energies are directed toward goals, resources, organization structures, or people, a manager is a problem solver. The manager asks: “What problems have to be solved, and what are the best ways to achieve results so that people will continue to contribute to this organization?” From this perspective, leadership is simply a practical effort to direct affairs; and to fulfill his or her task, a manager requires that many people operate efficiently at different levels of status and responsibility. It takes neither genius nor heroism to be a manager, but rather persistence, tough-mindedness, hard work, intelligence, analytical ability, and perhaps most important, tolerance and goodwill.

Another conception of leadership, however, attaches almost mystical beliefs to what a leader is and assumes that only great people are worthy of the drama of power and politics. Here leadership is a psychodrama in which a brilliant, lonely person must gain control of himself or herself as a precondition for controlling others. Such an expectation of leadership contrasts sharply with the mundane, practical, and yet important conception that leadership is really managing work that other people do.

TYPES OF LEADERSHIP STYLES

The seven primary leadership styles are:

- Autocratic Style.
- Authoritative Style.
- Pacesetting Style.
- Democratic Style.
- Coaching Style.
- Affiliative Style.
- Laissez-Faire Style.

IMPORTANT OF CONTEXT

Leader triangle

- Carrier
- Community
- Family

Learning to lead

- Performance
- Traits
- Hands on experience
- On job training

Tools for leading terms

- Positive Outlook
- Encouraging Attitude
- Recognition and Reward
- Open Door Policy
- In-House Promotion.

AFTERNOON SESSION DETAILS

Date:	**8-6-2020**	**Name:**	**Kavyashree m**
Course:	**Java**	**USN:**	**4al15ec036**
Topic:	**Introduction, Programming Core Java,Loops in java.**	**Semester & Section:**	**8th A**
Github Repository:	**kavya**		

Image of session

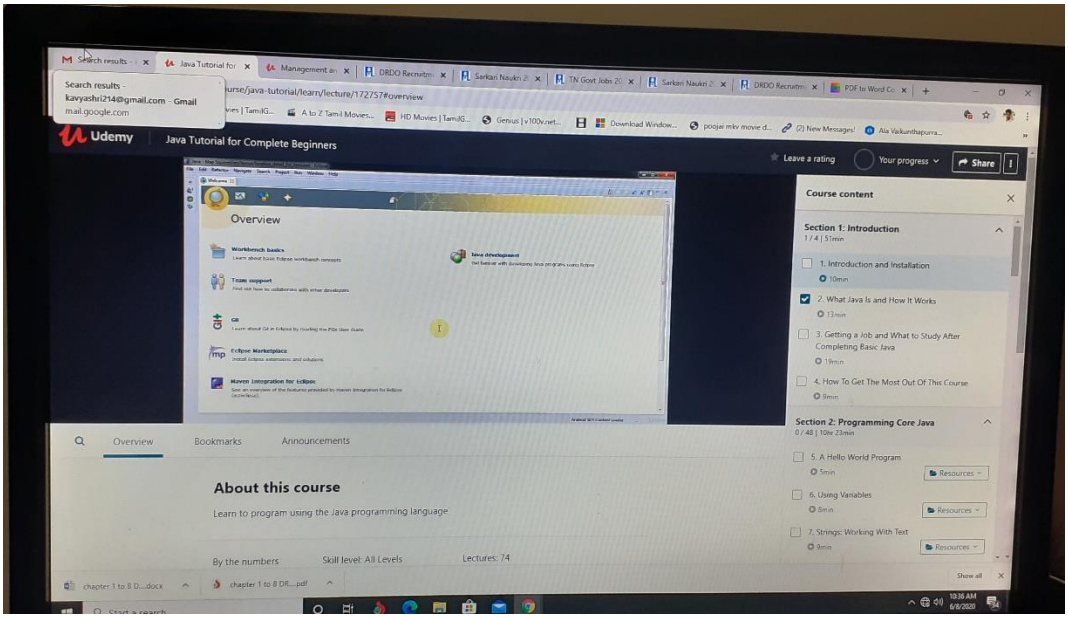


Fig 1: Java

Introduction

Java is a programming language created by James Gosling from Sun Microsystems (Sun) in 1991. The target of Java is to write a program once and then run this program on multiple operating systems.

Java is defined by a specification and consists of a programming language, a compiler, core libraries and a runtime (Java virtual machine) The Java runtime allows software developers to write program code in other languages than the Java programming language which still runs on the Java virtual machine. The Java platform is usually associated with the Java virtual machine and the Java core libraries.

The Java language was designed with the following properties:

- Platform independent: Java programs use the Java virtual machine as abstraction and do not access the operating system directly. This makes Java programs highly portable. A Java program (which is standard-compliant and follows certain rules) can run unmodified on all supported platforms, e.g., Windows or Linux.
- Object-orientated programming language: Except the primitive data types, all elements in Java are objects.
- Strongly-typed programming language: Java is strongly-typed, e.g., the types of the used variables must be pre-defined and conversion to other objects is relatively strict, e.g., must be done in most cases by the programmer.
- Interpreted and compiled language: Java source code is transferred into the bytecode format which does not depend on the target platform. These bytecode instructions will be interpreted by the Java Virtual machine (JVM). The JVM contains a so called Hotspot-Compiler which translates performance critical bytecode instructions into native code instructions.
- Automatic memory management: Java manages the memory allocation and de-allocation for creating new objects. The program does not have direct access to the memory. The so-called garbage collector automatically deletes objects to which no active pointer exists.

Hello world Java program

// a small Java program

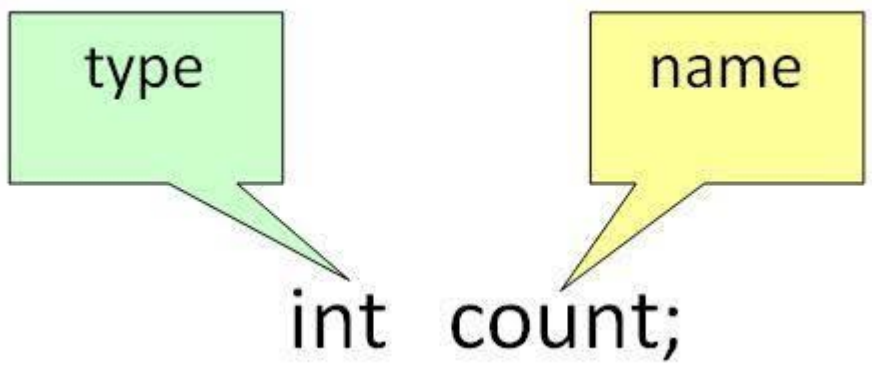
```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World");
   }
```

Using variable

A variable is a name given to a memory location. It is the basic unit of storage in a program.

- The value stored in a variable can be changed during program execution.
- A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.
- In Java, all the variables must be declared before use.

How to declare variables?



We can declare variables in java as follows:

type: Type of data that can be stored in this variable.

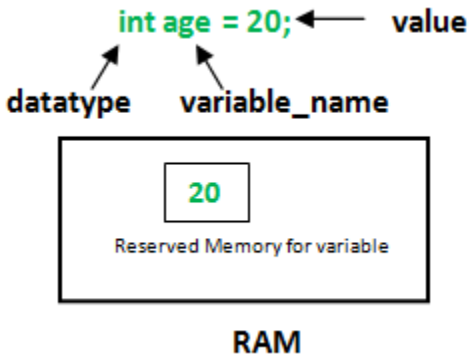
name: Name given to the variable.

In this way, a name can only be given to a memory location. It can be assigned values in twoways:

Variable Initialization

- Assigning value by taking input

How to initialize variables?



Types of variables

There are three types of variables in Java:

- Local Variables
- Instance Variables
- Static Variables

Let us now learn about each one of these variables in detail.

1. **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:

filter_none

edit

play_arrow

brightness_4

```
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();
    }
}
```

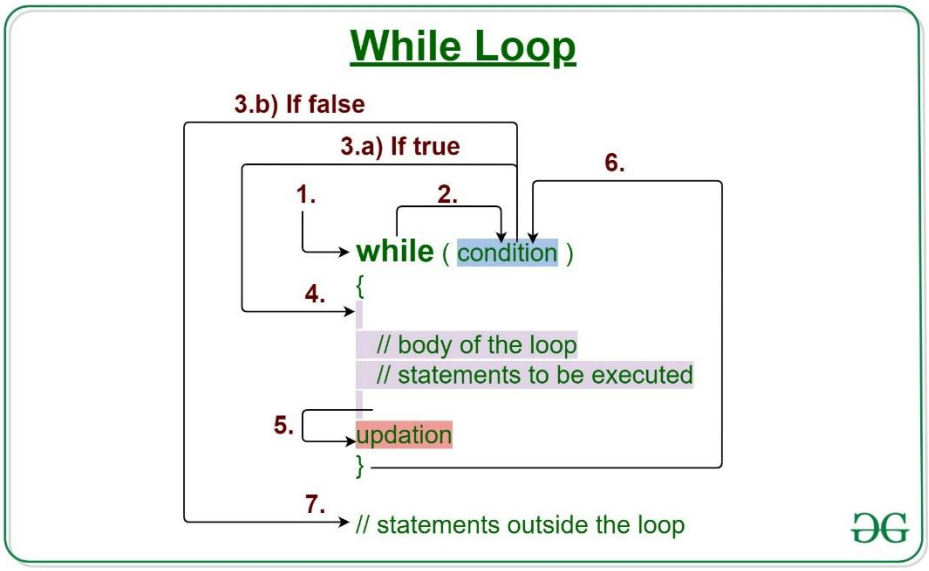
Output: Student age is : 5

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.

While loop

Java while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.



Syntax:

```
while (test_expression)
{
  // statements


 update_expression;
}
```

The various parts of the While loop are:

1. Test Expression: In this expression we have to test the condition. If the condition evaluates to true then we will execute the body of the loop and go to update expression. Otherwise, we will exit from the while loop.

   Example:

i <= 10

2. Update Expression: After executing the loop body, this expression increments/decrements the loop variable by some value.

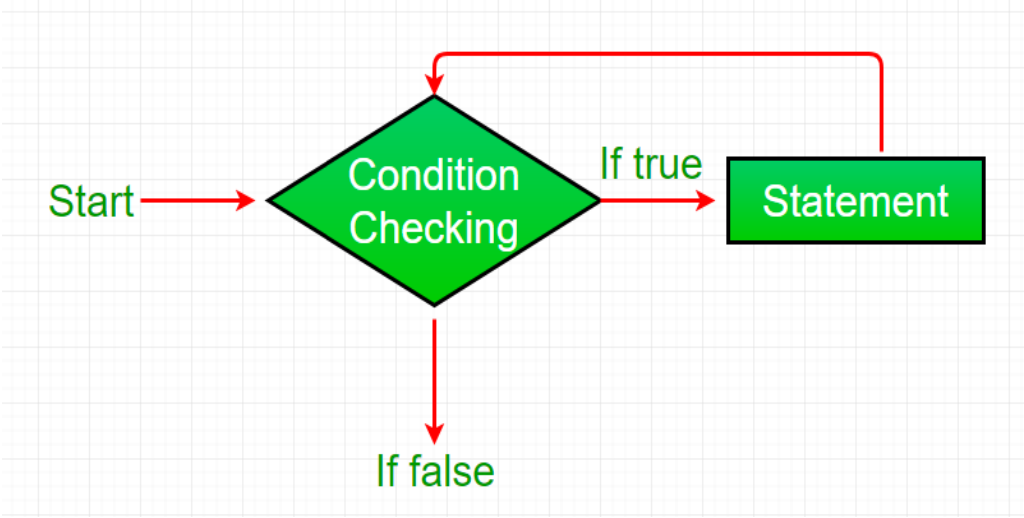
   Example:

i++;

How does a While loop executes?

1. Control falls into the while loop.
2. The flow jumps to Condition
3. Condition is tested.
   a. If Condition yields true, the flow goes into the Body.
   b. If Condition yields false, the flow goes outside the loop
4. The statements inside the body of the loop get executed.
5. Updation takes place.
6. Control flows back to Step 2.
7. The do-while loop has ended and the flow has gone outside.

Flow chart while loop (for Control Flow):



Example 1: This program will try to print “Hello World” 5 times.

filter_none

edit

play_arrow

brightness_4

```
// Java program to illustrate while loop.

class whileLoopDemo {
    public static void main(String args[])
    {
        // initialization expression
        int i = 1;

        // test expression
        while (i < 6) {
            System.out.println("Hello World");

            // update expression
            i++;
        }
    }
}
```

Output:

```
Hello World
Hello World
Hello World
Hello World
Hello World
```

For loop in java

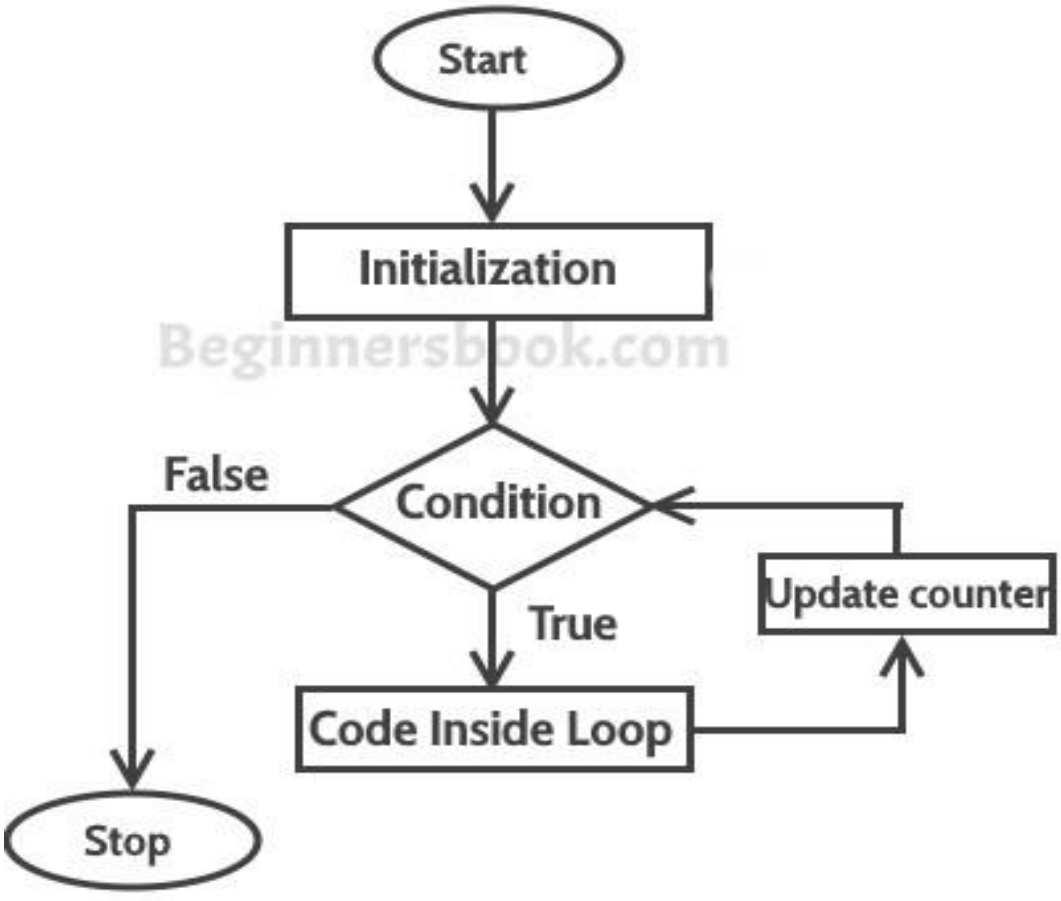
Loops are used to execute a set of statements repeatedly until a particular condition is satisfied. In Java we have three types of basic loops: for, while and do-while. In this tutorial we will learn how to use “for loop” in Java.

Syntax of for loop:

```
for(initialization; condition ; increment/decrement)
{
  statement(s);
}
```

Flow of Execution of the for Loop

As a program executes, the interpreter always keeps track of which statement is about to be executed. We call this the control flow, or the flow of execution of the program.



If statement in java

Java if Statement

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

      int number = 10;

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

Output:

```
The number is positive.
This statement is always executed.
```