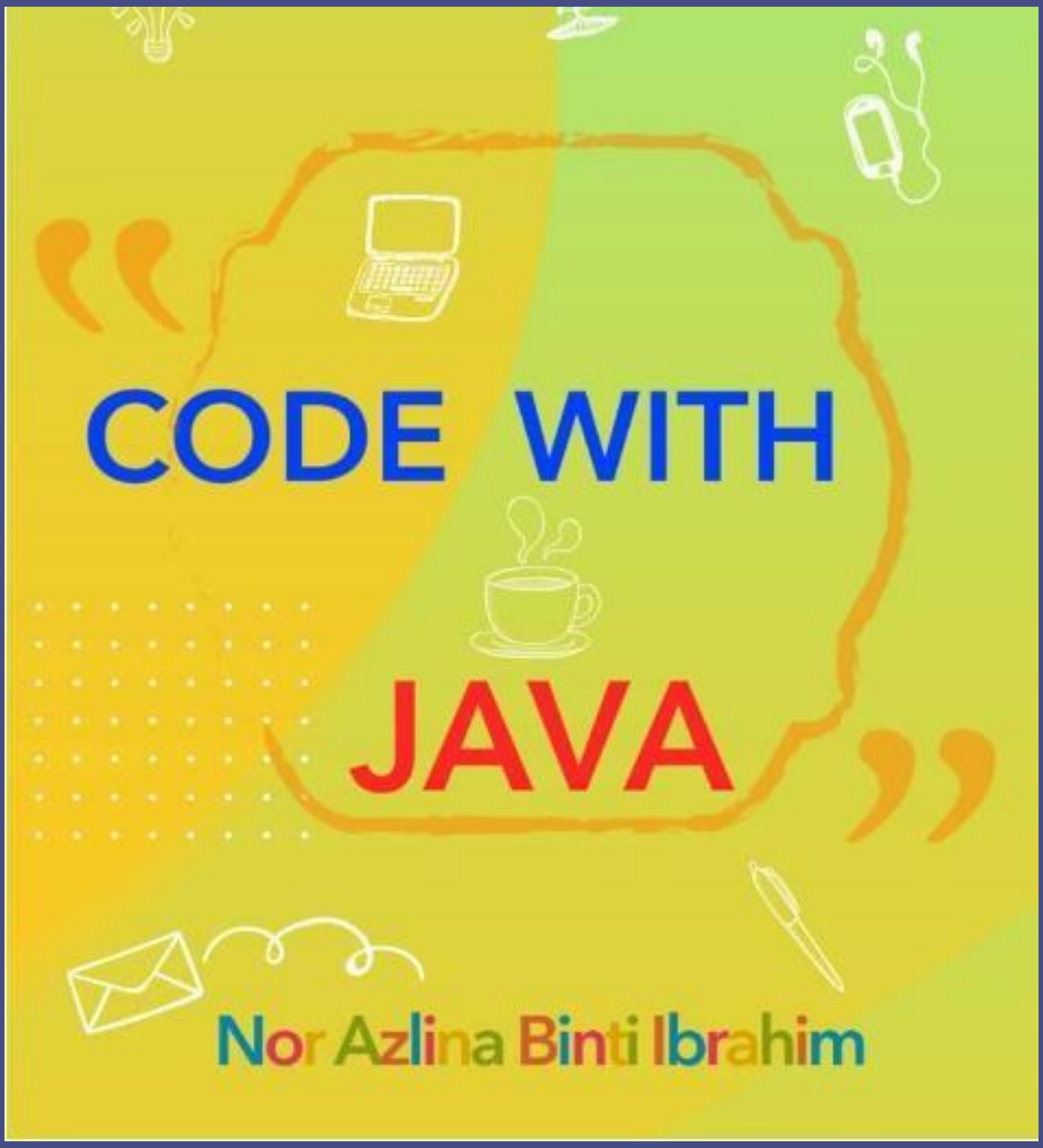
CODE WITH
JAVA
Nor Azlina Binti Ibrahim



Jabatan Teknologi Maklumat Dan Komunikasi
Politeknik Sultan Mizan Zainal Abidin
KM 8, Jalan Paka
23000 Dungun, Terengganu

Code With JAVA

Table of Contents

1 Fundamentals of Java Programming

15 Control Structure And Array

25 Classes, Objects And Methods

32 Inheritance And Polymorphism

45 Exception Handling

50 String Manipulation And Multithreading

Preface

CODE WITH JAVA – gets you started programming in Java. This book allow you to begin with the basics, how to create, compile, and run a Java program. It also includes example code with commentary that describes the programming techniques.

Highlights -

- Fundamentals of Java Programming
- Control Structure And Array
- Classes, Objects And Methods
- Inheritance And Polymorphism
- Exception Handling
- String Manipulation and Multithreading

Abstract

Code With JAVA
Edition 2023

Published by:
Politeknik Sultan Mizan Zainal Abidin.
KM 08, Jalan Paka, 23000 Dungun, Terengganu Darul Iman.
Tel: 09-8400800 Fax: 098458781
www.psmza.edu.my

Copyright © 2023 @zlin@
All rights reserved. No part of this document may reproduced,stored in retrieval system or transmitted in any form or by any means (electronic, mechanical, photocopying recording or otherwise) without the permission of the copyright owner.

CHAPTER 1

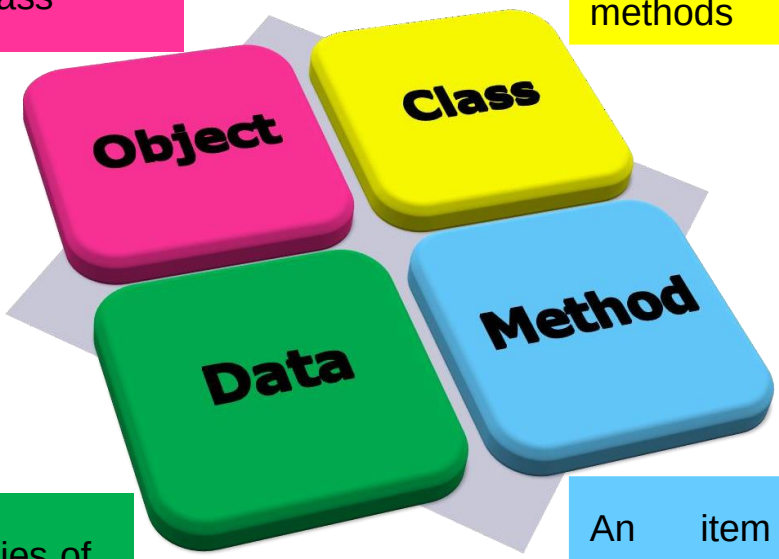
Fundamentals of Java Programming

OOP

Programming paradigm using objects, data structures consisting of data fields and methods together with their interactions to design applications and computer programs.

An instance of a particular class

set of object consisting similar attributes and methods



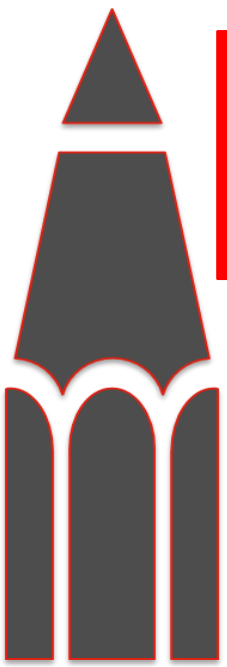
The properties of object

An item that can manipulate objects

Inheritance
Creating a new class based on an initial class, with similar attributes

Data abstraction
Process of removing all unimportant attributes

Polymorhism
process of delivering the same message to two or more different objects, and it produces different actions



Encapsulation
Process of binding data and method that forms a class and controls data access with information hiding.

Concept of OOP

JAVA

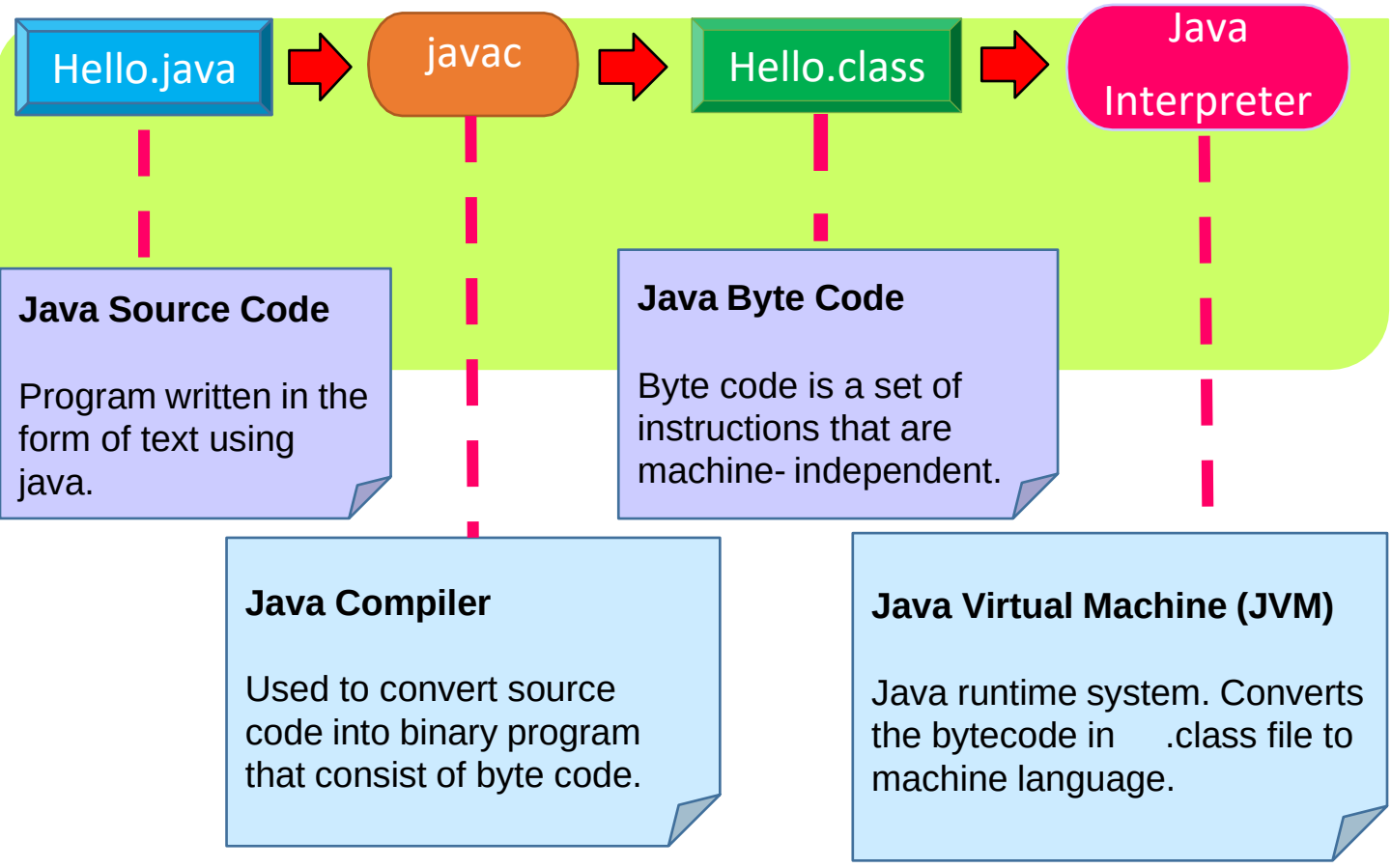
To develop Web application, animation and interactive interface.

Can also develop stand-alone program for software application development.

Java Development Kit (JDK)

Contains the software and tools that you need to compile, debug, and run applets and applications that you've written using the Java programming language.

Java Architecture



JAVA PROGRAM

Executing java program

1. Install JDK
2. Set path of the jdk/bin directory
3. Create the java program
4. Compile and run the java program

To compile your program
javac class_name.java

To run your program
java class_name

Single-line comment(//).
Multiple-line comment(/*…*/).

Class definition
Greeting is an identifier that is the name of the class

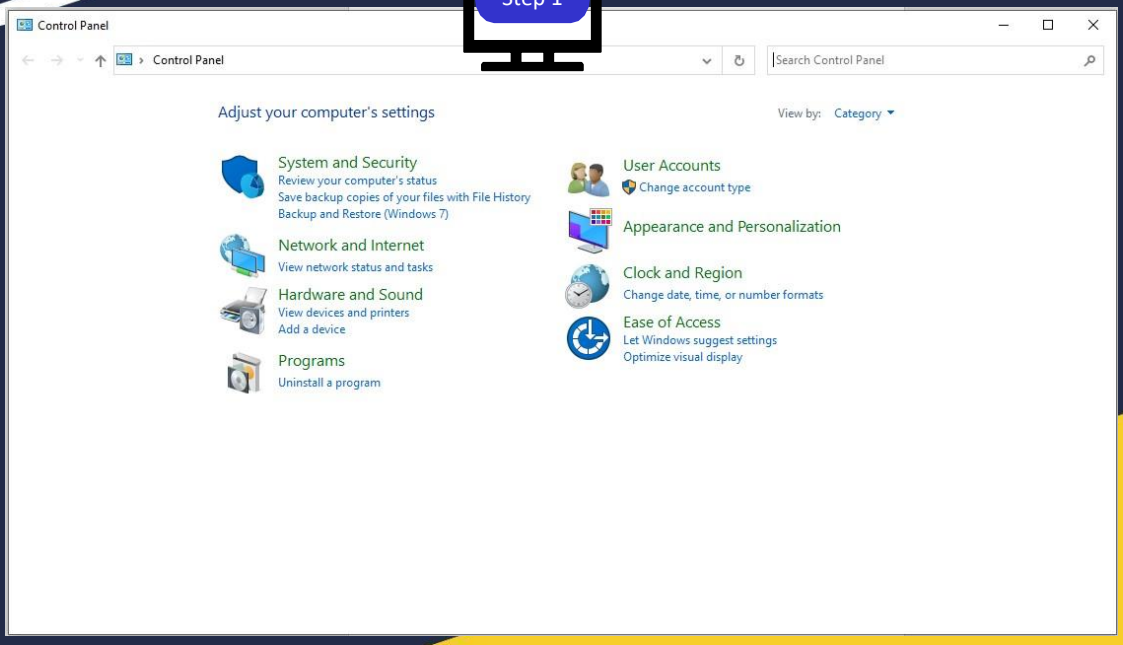
```
//comment
class Greeting
{
    public static void main (String args[ ])
    {
            System.out.println("Welcome to Java");
    }
}
```

main method: This is the main method. Every application in Java must contain a main method.

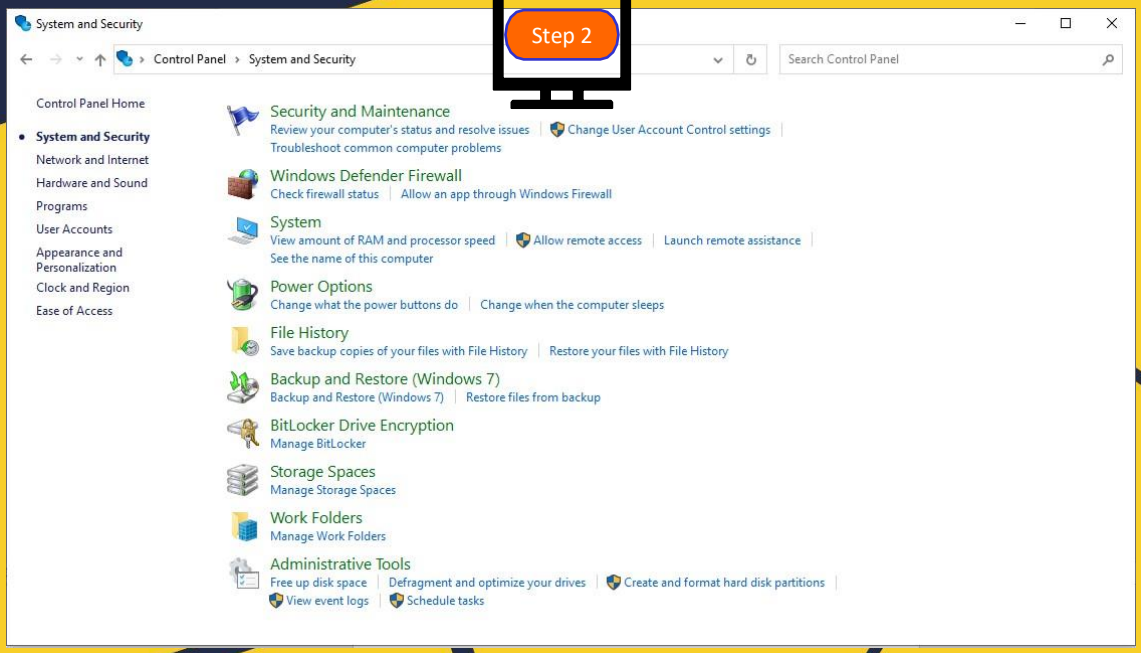
The following code prints the string inside quotation marks Hello, World! to standard output (your screen).

HOW TO SET PATH OF JDK IN WINDOWS

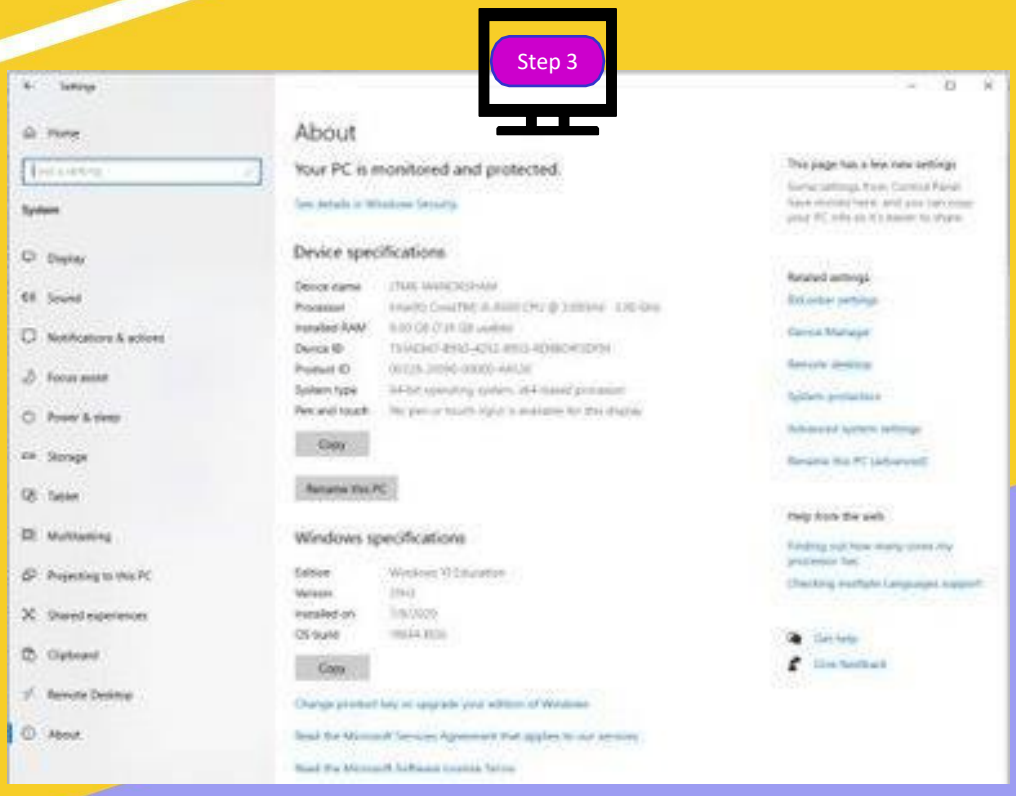
Step 1

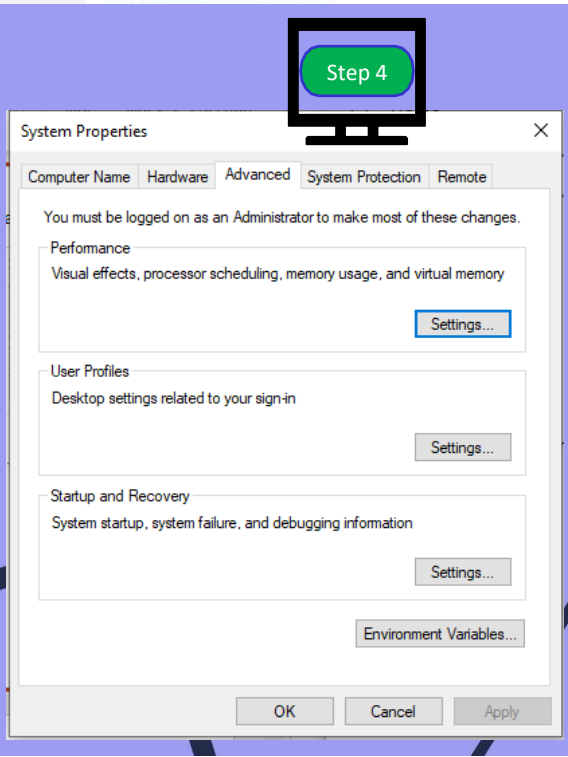


Step 2

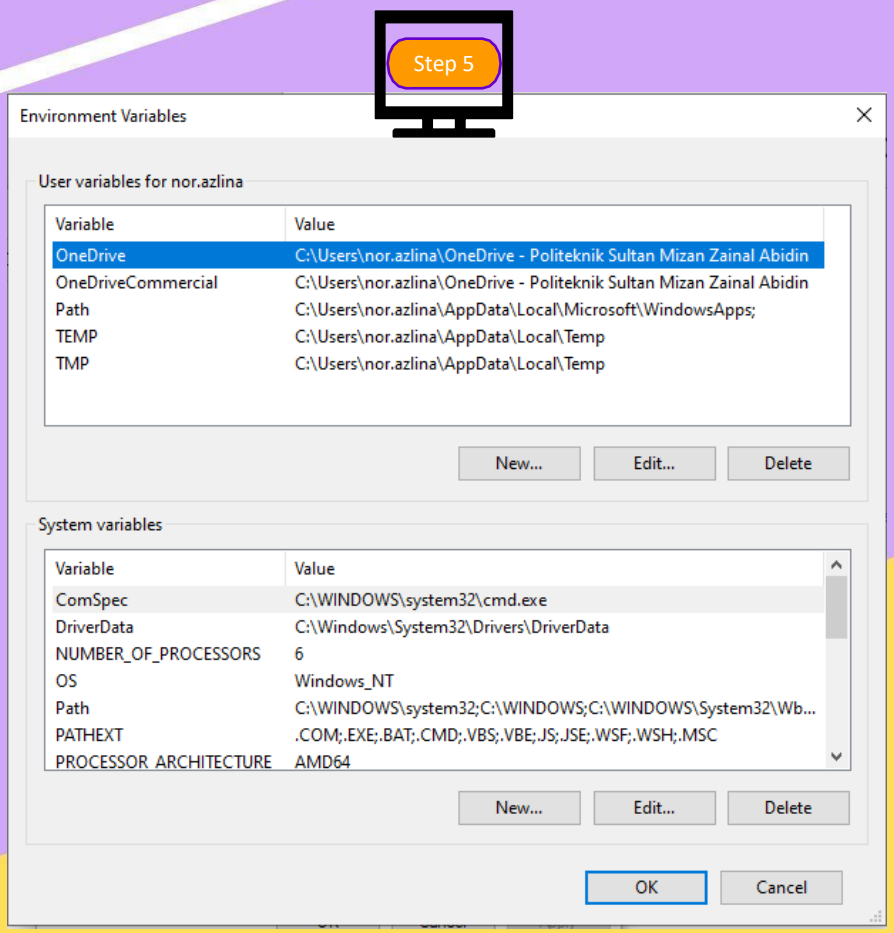


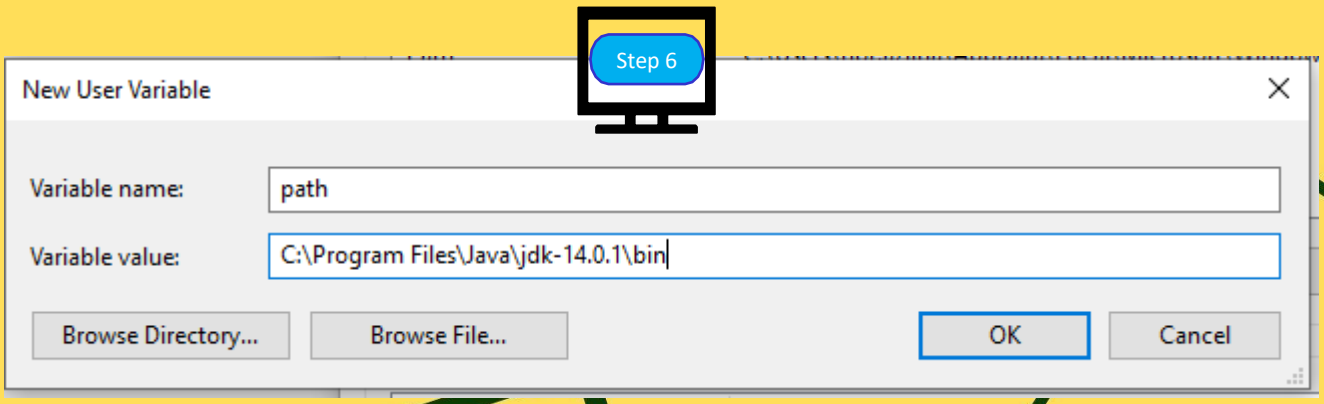
HOW TO SET PATH OF JDK IN WINDOWS





HOW TO SET PATH OF JDK IN WINDOWS

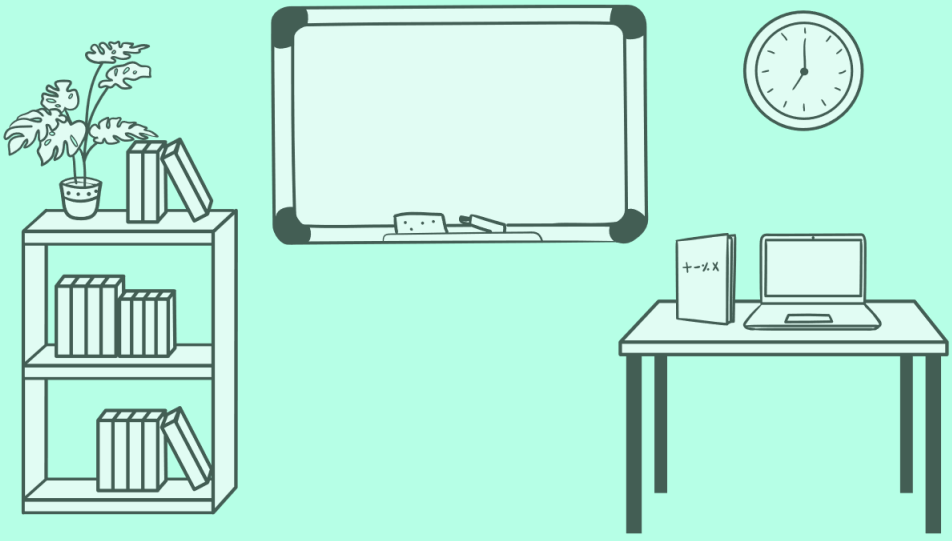




Set path of JDK

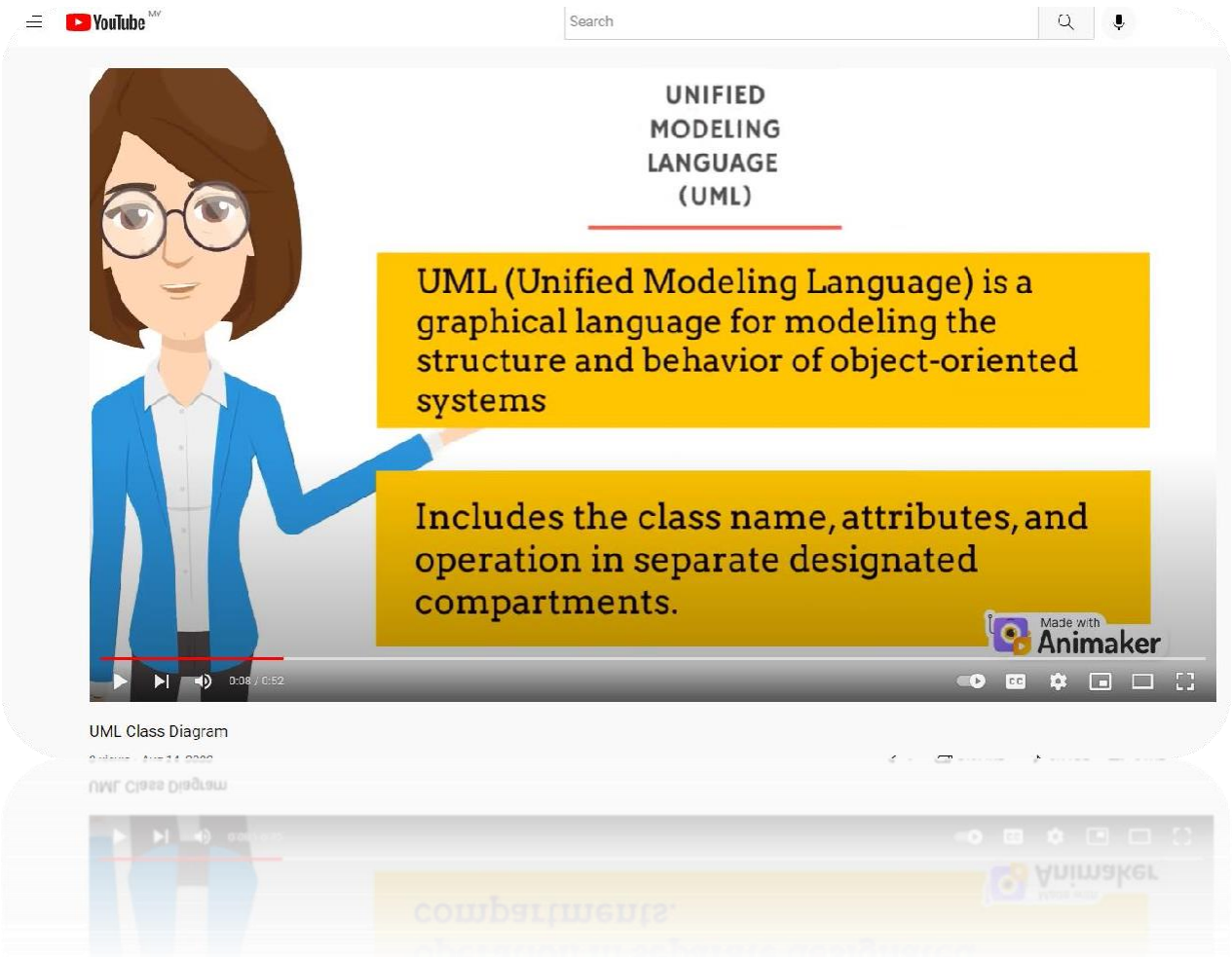
WATCH VIDEO

https://youtu.be/kuogy1bhdfY



Unified Modeling Language (UML)

https://youtu.be/r3a7c



Object Oriented Analysis And Design (OOAD)

To emphasize a problem domain and logical solution from the perspective of objects (things, concepts or entities)

OO Analysis

- To find and describe the objects or concepts in the problem domain.
- For example in the Library Information System, some of the concepts include Book, Library and Patron.

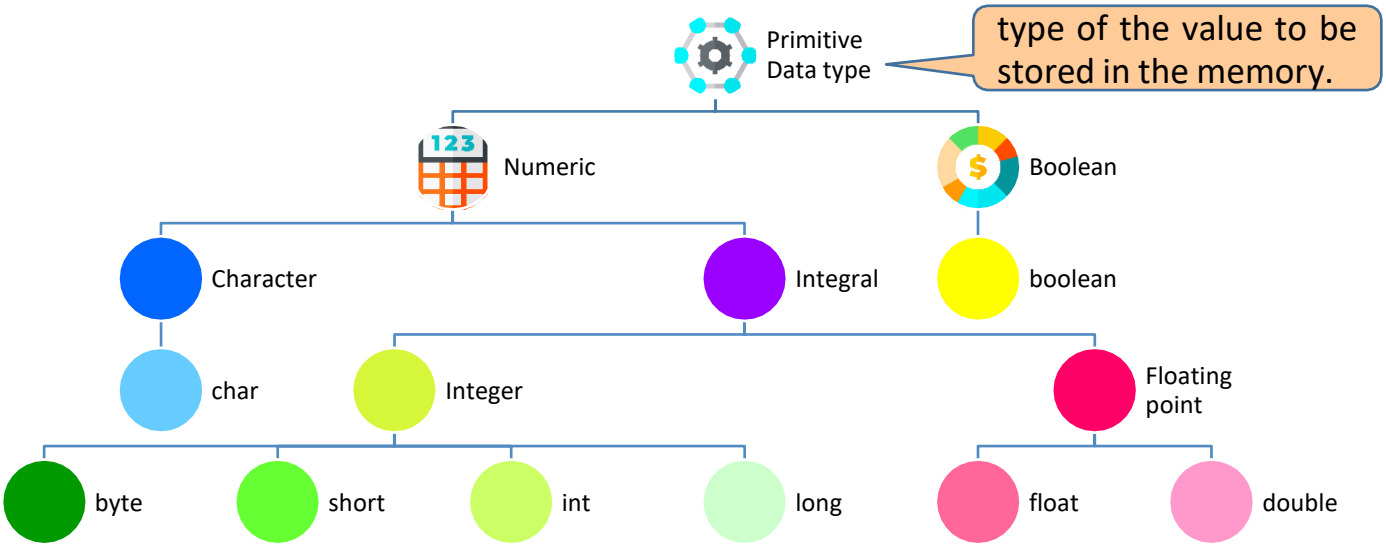
OO Design

- To define software objects (attributes and operations) and how they collaborate to fulfill the requirements that will be implemented in an OO programming language.
- For example in the Library Information System, a Book is an object that have a title as it attribute and getChapter as the method.

OO Programming

- Design components are implemented.

Data Types,Variables And Constants



Variables

Memory location where a value is stored.

Syntax
data type variable_name;

Example
int mark=85;
double num1, num2;
char gender='F';

Constant

Value that does not change throughout program execution

Syntax
final data type constant_name = value;

Example
final double PI=3.14;

Formatting Output

Syntax:
System.out.printf(format,variable);

Example:
int count=7;
double total=50.77;
System.out.printf("Count is %d and total is %f", count, total);

Specifier	Output	Example
%b	A boolean type	True or false
%c	A character	'a'
%d	A decimal integer	200
%f	A floating point number	17.89
%s	A string	"I love Java"

Sample Program

Variable And Constant

Example 1 — Variable And Constant

```
class circle
{
public static void main(String args[])
{
        int radius=3;
        final double PI=3.142;
        double area;
        area=PI*radius*radius;
        System.out.println(area);
System.out.println("The area of the circle is:"+area);
System.out.printf("The area of the circle is: %.2f",area);
}
}
```

Variable

Constant

Input Stream And Output Stream

refers to the flow of data to a program from an input device **(System.in).**

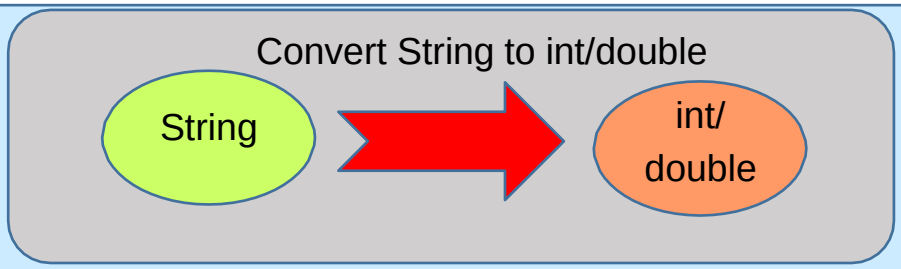
refers to the flow of data from a program to an output device **(System.out).**

This is a package to be used while using input and output statements in a program

This is to enable input from keyboard

```
import java.io.*;
class masukan
{
  public static void main (String args[]) throws IOException
 {
 BufferedReader masuk = new BufferedReader(new InputStreamReader(System.in));
System.out.print("Enter your name: ");
String name = masuk.readLine();
System.out.print("Enter your reg num: ");
String num = masuk.readLine();
System.out.println("Name: " + name);
System.out.println("Reg Num: " + num);
  }
}
```

Text data is received from the standard input device, keyboard



Data is received as a string. If entered data is in number format, we need to convert the string to an int/double. To do so, we use

Integer.parseInt()

Double.parseDouble()

Sample Program

Input Stream And Output Stream

```
import java.io.*;
class masukan
{
    public static void main (String args[]) throws IOException
    {
    BufferedReader masuk = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("Enter your name: ");
    String name = masuk.readLine();
    System.out.print("Enter your reg num: ");
    String num = masuk.readLine();
    System.out.println("Name: " + name);
    System.out.println("Reg Num: " + num);
    }
}
```

This is a package to be used while using input and output statements in a program

This is to enable input from keyboard

Text data is received from the standard input device, keyboard

CHAPTER 2

Control Structure And Array

Selection Statement

A programming language uses control statements to control the flow of execution of program based on certain conditions.

if

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

if - else

```
if(condition){
// if condition is true
}else{
//if condition is false
}
```

else if

```
if(condition1){
//if condition1 is true
}else if(condition2){
// if condition2 is true
}
else{
// if all the conditions are false
}
```

nested if

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

switch

```
switch(expression){
case value1:
 //code to be executed;
 break;
case value2:
 //code to be executed;
 break;
default:
 code to be executed if all cases
 are not matched;
}
```

A break statement passes the control outside switch structure.

Sample Program

Selection statements

If - else

Executes the set of statements in if block, when the given condition is satisfied.

Executes the statements in the else block, when the condition is not satisfied.

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

 public static void main (String args[]) throws IOException {
 BufferedReader masuk = new BufferedReader(new InputStreamReader(System.in));

int num;

System.out.print("Enter the num: ");
String indata = masuk.readLine();

num = Integer.parseInt(indata);

 if (num<0)
        {
        System.out.println("Negative");
        }
 else
        {
        System.out.println("Positive");
        }
}
```

Sample Program

Selection statements

Switch case

The case statements are executed based on the value of the expression.

A break statement passes the control outside switch structure.

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

public static void main(String args[]) throws IOException {
BufferedReader masuk = new BufferedReader(new
InputStreamReader(System.in));

 System.out.print("Enter the month: ");
 String indata = masuk.readLine();

 int month=Integer.parseInt(indata);

    switch (month)
     {
      case 1:     System.out.println("January");
         break;
      case 2:     System.out.println("February");
         break;
      case 3:     System.out.println("March");
          break;
       default: System.out.println("wrong choice");
    }
}
```

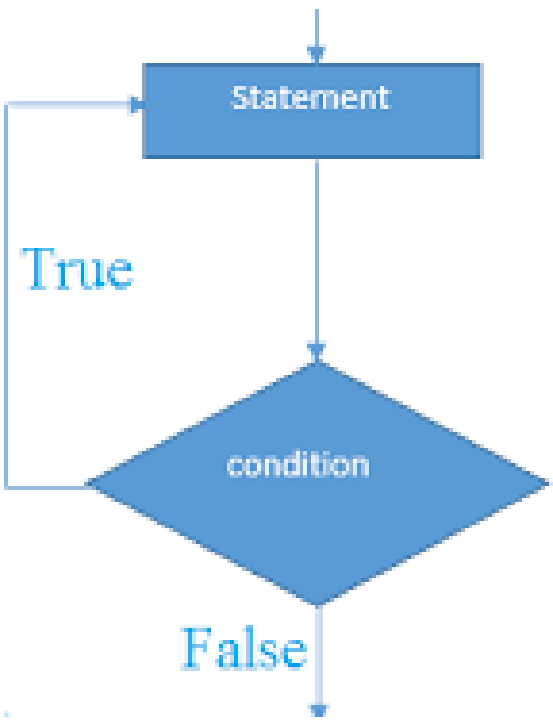
Looping Statement

for

```
for (initialisation; condition; incrementation/decrementation)
  {
    /loop statements
  }
```

while

```
while(condition)
  {
     //loop statements
     Increment/decrement;
  }
```



do..while

```
do
   {
      //loop statements
   } while(condition);
```

break statement causes termination of the loop and transfers the control outside the loop.

continue statement will transfer the control to the beginning of the loop

For

Looping statement enables you to repeat a set of instructions based on the condition.

```
class cthloop {

public static void main(String a
   int value =3;
   int num,multiple;
   System.out.println("Multiples of 3:\n");

   for (num = 1; num <= 10; num++)
      {
       multiple=num*value;
          System.out.println(+multiple);
      }
}
}
```

For

Initialisation: sets a variable before the loop starts

Condition: If the condition is true, the loop will start over again, if it is false, the loop will end

Incrementation / Decrementation: increases/decrease a value

Sample Program

Looping Statement

Do..While

Do..While executes the statements first before evaluating the condition of the loop

```
class cthdo {

public static void main(String args[]) {
        int x=20;

  do
        {
        System.out.print(x);
              x=x+1;
        } while(x<15);
    }

}
```

While

The while loop evaluates the condition at the beginning of the loop before the statement

```
class cthwhile {

public static void main(String args[]) {
      int sum=0;
      int j=0;

  while(j<10)
        {
        sum=sum+j;
        j++;
        }
System.out.println("The total of first 10 numbers is: " +sum);
    }
}
```

Sample Program

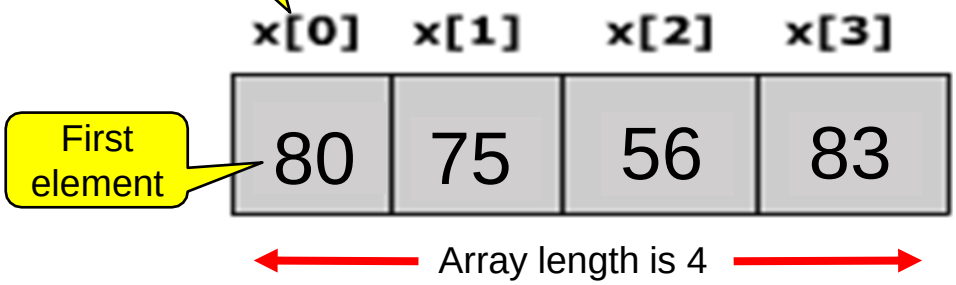
Looping Statement

Array 1D

Data type Variable_name[ ] = new Data type [Size of the array]

Example: int marks[ ] = new int[4];

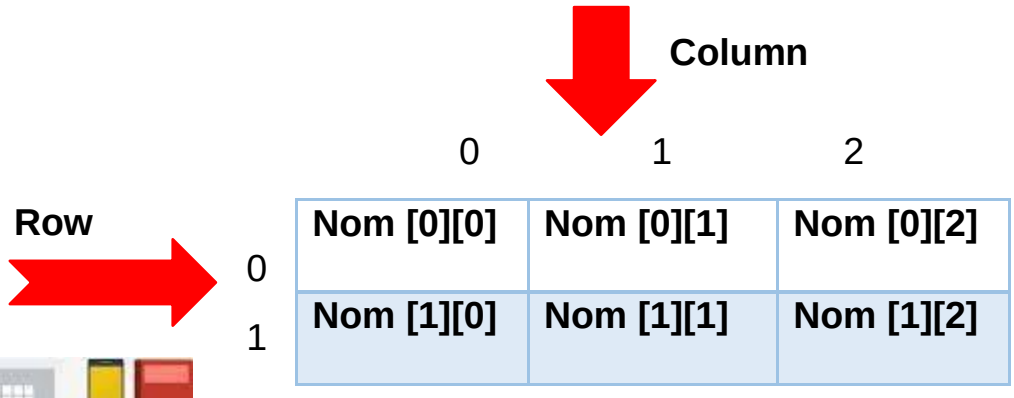
Index start from 0



Array 2D

Data type Variable name[][] = new Data Type [Row][Column];

Example:
int nom[][] = new int[2][3]; //2 row and 3 column



Array

collection of similar type of variables having a common name.

Sample Program

Array

Array 1D

Will have only one subscript [ ]. Subscript refers to the dimension of the array.

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


      int marks[ ] = new int[5];

      marks[0]=95;
      marks[1]=86;
      marks[2]=75;
      marks[3]=80;
      marks[4]=66;

      int i;

      for (i=0; i<=4; i++)
      {

System.out.println("Marks "+(i+1)+":" + marks[i]);

      }
}
}
```

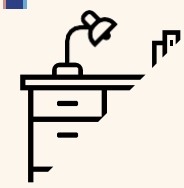
Array declaration

Array Initialisation

Array Acessing

Sample Program

Array



Array 2D

Two-dimensional arrays represent data in terms of rows and columns. It has two subscripts

```
class tatasusunan2d {
public static void main(String args[])  {

int[ ][ ] Numbers = new int[3][2];

Numbers[0][0] = 10;
Numbers[0][1] = 12;

Numbers[1][0] = 20;
Numbers[1][1] = 45;

Numbers[2][0] = 30;
Numbers[2][1] = 67;

int rows = 3;
int columns = 2;

int i, j;

for (i=0; i < rows ; i++) {

      for (j=0; j < columns ; j++) {

      System.out.print(Numbers[i][j] + " " );

      }
System.out.println( "" );

}
}
}
```

10	12
20	45
30	67

CHAPTER 3

Classes, Objects And Methods

Method

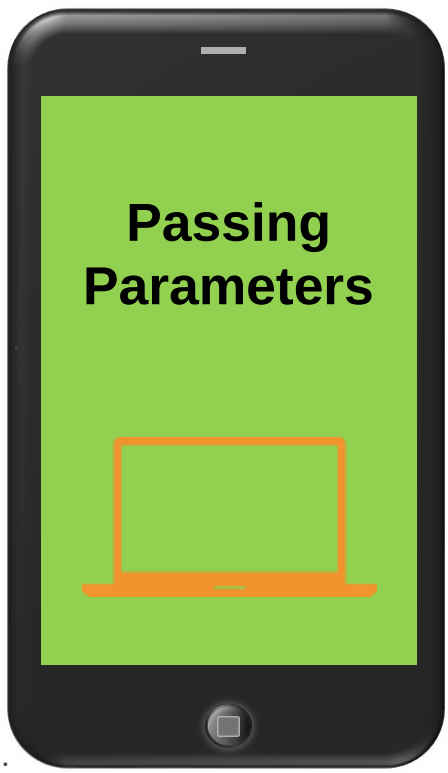
Is a component of the program written to perform a specific task.

A program can be split into number of smaller programs called modules.

```
Syntax
return datatype  method_name (parameter list)
{
  Statements;
}
```

Pass-by-value

Pass-by-reference



Only the value of the variable is passed to the method.

Any change made to the variable in the called method will **not affect** the variable in the calling method.

The address of the variable is passed to the method

Any change made to this variable in the called method **will affect** the variable in the calling method.

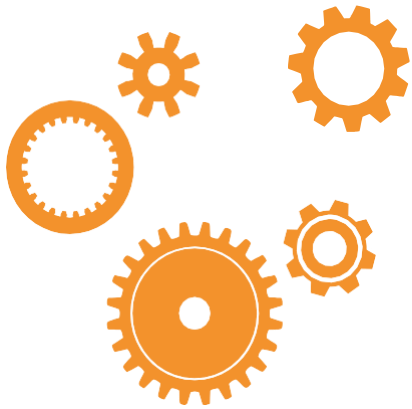
Sample Program - Method

```
public class cth_method {

   public static void main(String[] args) {
      int a = 12;
      int b = 6;
      int hasil_tambah = tambah(a, b);
      System.out.println("Hasil tambah = " + hasil_tambah);
      int hasil_tolak=tolak();
      System.out.println("Hasil tolak = " + hasil_tolak);
      bahagi(a,b);
      darab();  }

   public static int tambah(int n1, int n2) {
       int jum;
       jum=n1+n2;
       return jum;   }

   public static int tolak() {
      int x=10;
      int y=6;
      int beza=x-y;
      return beza;   }

   public static void bahagi(int nom1, int nom2) {
    int hasil_bahagi=nom1/nom2;
    System.out.println("Hasil bahagi = " + hasil_bahagi);   }

   public static void darab() {
    int i=7;
    int j=3;
    int hasil_darab=i*j;
    System.out.println("Hasil darab = " + hasil_darab);
   }
}
```

Object & Constructor

Object

- An entity that consists of data and methods.
- Object is a member of a class.
- Class can have many objects.
- An object has a unique identity, state and behavior.
- The state of an object consists of a set of data fields or properties.
- The behavior of an object is defined by a set of methods.



```
ClassName  ObjectName = new ClassName( );
```

Example:

```
Book  IT = new Book ();
```

Constructor

- Special kind of method.
- Must have the same name as the class itself
- Do not have a return type
- Play the role of initializing objects.

SAMPLE PROGRAM
OBJECT & CONSTRUCTOR

Example 1

classes and objects

```
class boxoneclass
{
   public static void main(String arg[])

      {
      int width = 10, height = 10, depth = 10;

      boxoneclass squareBox = new boxoneclass();

       int volume;

      volume = squareBox.ComputeVolume(width,height,depth);
      System.out.println("The volume of the box  is "+volume);

      }

          int ComputeVolume(int w,int h,int d)
          {
                    int isipadu;
                    isipadu= w*h*d ;
                    return isipadu ;
          }
}
```

Create object

Object.method

SAMPLE PROGRAM
OBJECT & CONSTRUCTOR

Example 2

constructor

```
class boxcon1
{
   public static void main(String arg[])
   {
      boxcon1 squareBox = new boxcon1(10,10,10);

     int volume;
     volume = squareBox.ComputeVolume();
     System.out.println("The volume of the box  is "+volume);
   }

      int width,height,depth;
      int ComputeVolume()
      {
      return (width*height*depth);
      }

   boxcon1(int w,int h,int d)
            {
            width=w;
            height=h;
            depth=d;
             }

}
```

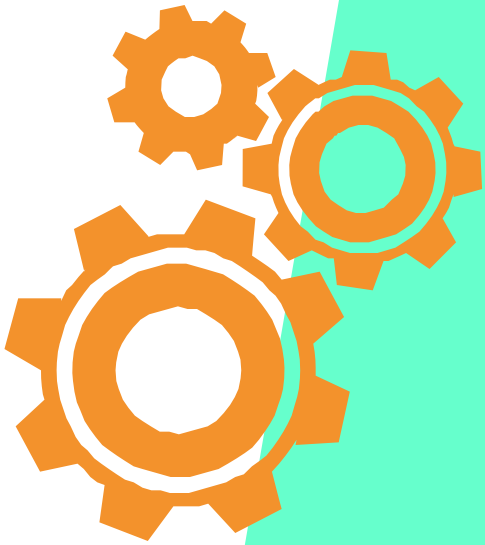
Constructor

OBJECT & CONSTRUCTOR

https://youtu.be/-VWcCPUypEI

CHAPTER 4

Inheritance And Polymorphism



Process of delivering the same message

Produces different actions

Polymorphism

Using techniques:

- Method Overloading

- Constructor Overloading.

- ❑ Two methods which have same name but different parameters
- ❑ parameters in overloaded methods should differ in at least one of the following:
  - Number of parameters
  - Data type of the parameters

Sample Program
-Polymorphism

```
class Product
{
 public static void main(String args[])
 {
   int result1;
   double result2;
   prod obj=new prod();

   result1=obj.multiply(12,13);
   result2=obj.multiply(2.5,3.2);

   System.out.println(result1);
   System.out.println(result2);
 }
}

class prod
{
   int multiply(int No1, int No2)
    {
      return(No1*No2);
    }

   double multiply (double No1,
   double No2)
    {
      return(No1*No2);
    }

}
```

Method Overloading

Sample Program - Polymorphism

```
class conover
{
   public static void main(String arg[])
   {
      conover obj1 = new conover(10,10,10);
      conover obj2 = new conover(10,10);
     int volume;
     volume = obj1.ComputeVolume();
     System.out.println("The volume of the box is "+volume);
      int area;
      area = obj2.ComputeArea();
     System.out.println("The area of the box is "+area);
  }
      int width,height,depth;
      int ComputeVolume()
      {
      return (width*height*depth);
      }
int ComputeArea()
      {
      return (width*height);
      }
```

```
conover(int w,int h,int d)
        { width=w;
        height=h;
        depth=d;}
```

```
conover(int w,int h)
        { width=w;
        height=h;}
}
```

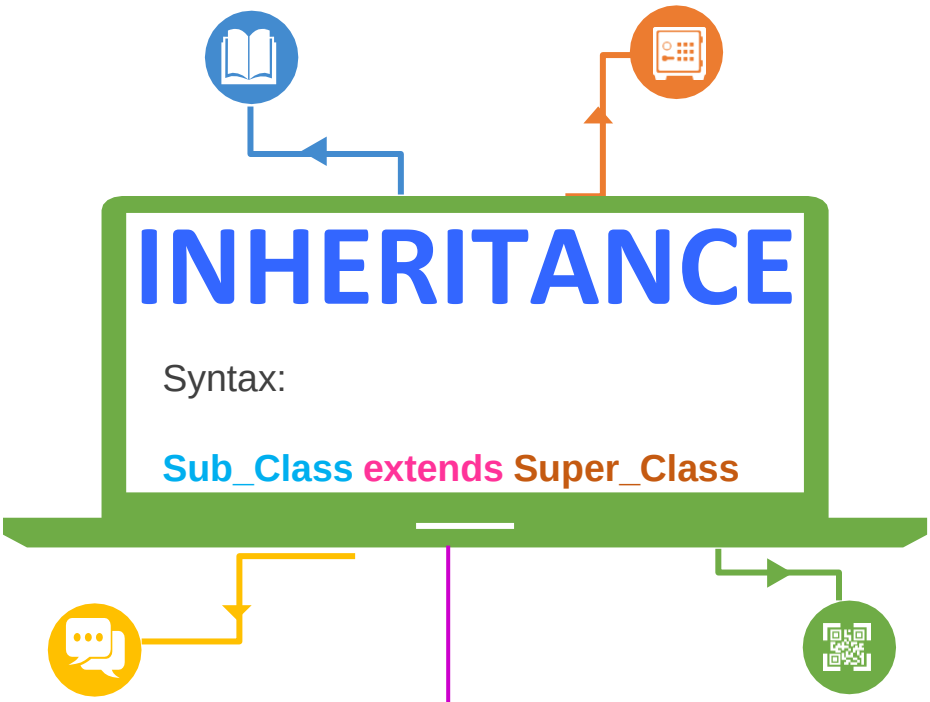
Constructor Overloading

Definition

The process by which one class acquires the properties(data members) and functionalities(methods) of another class.

Super Class

- The existing class from which a new class is created.
- It is also called as *Base class*



Sub Class

- The new class that is inherited from an existing class
- This class is also called as *Derived class*.
- The subclass inherits all the accessible methods and member variables of the super class.
- A subclass is created using the *extends* keyword.

THIS & SUPER Keyword

- Keyword THIS is a reference variable in Java that refers to the current object.
- The SUPER keyword refers to the objects of immediate parent class

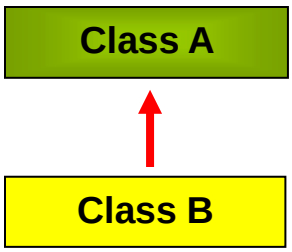
Method Overriding

- Concept of the methods in the sub class and the super class using the same method name with identical signature.

Types of Inheritance

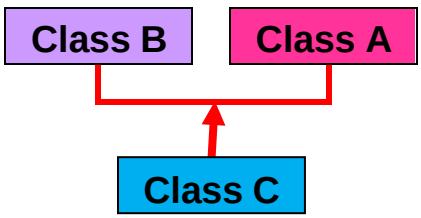
Single Inheritance

A super class can have any number of sub classes but a sub class can have only one super class.



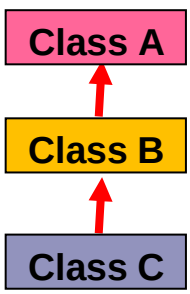
Multiple Inheritance

- ✓ A class is inherited from more than one existing class
- ✓ Java supports Multiple Inheritance through a concept called Interface.



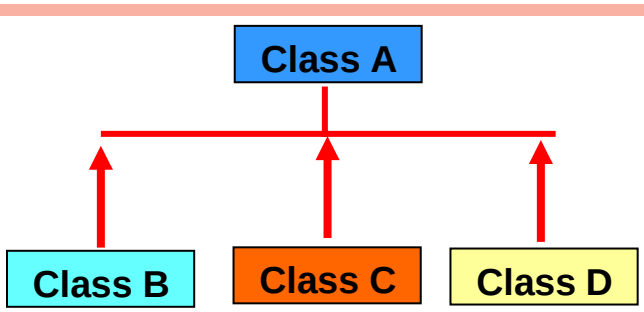
Multi Level Inheritance

- ✓ The extension of Single Inheritance.
- ✓ The sub class at the lowest level can access the member variables and methods of all the super classes at the higher level.



Hierarchical Inheritance

- ✓ When more than one classes inherit a same class



Hybrid Inheritance

- ✓ Combination of more than one types of inheritance in a single program

Sample Program - Inheritance

```
class Satu
{

int a;
float b;

        void Show()
        {
        System.out.println("b in super class:  " + b);
        }
}

class Dua extends Satu{

int a;
float b;

        Dua( int p, float q)
        {
        a = p;

        }


        void Show()
        {

        System.out.println("b in super class:  " + super.b);
        System.out.println("a in sub class:     " + a);
        }

        public static void main(String[] args)
        {
        Dua subobj = new Dua(1, 5);
        subobj.Show();
        }
}
```

Method Overriding

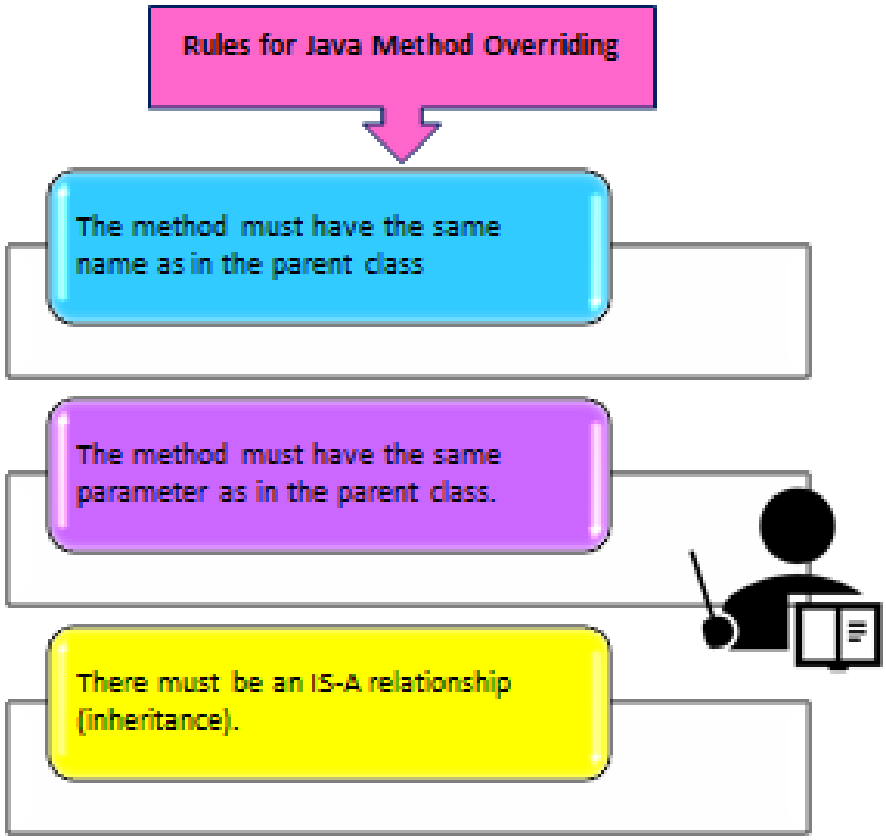
Method overriding refers to the concept of the methods in the sub class and the super class using the same method name with identical signature.

Object of super class

Method super class implemented

Object of sub class

Method sub class implemented



SAMPLE PROGRAM

METHOD OVERRIDING

```
class SuperA
{
        int i;
        SuperA(int a, int b)
        {
        i = a-b;
        }

void minus()
{
System.out.println("Total after subtract a and b is : " + i);
}
}

class SubClassB extends SuperA
{
        int j;
        SubClassB(int a, int b, int c)
        {
        super(a, b);
        j = a-b-c;
        }

void minus()
{
super.minus();
System.out.println("Total after minus a, b and c is : " + j);
}
}

class MethodOverriding
{
        public static void main(String args[])
        {
        SubClassB b = new SubClassB(30, 20,  10);
        b.minus();
        }
}
```

Abstract

- Abstract classes are like regular classes with data and methods, but you cannot create instances of abstract classes using the new operator.
- An abstract method is a method signature without implementation. Its implementation is provided by the subclasses.
- A class that contains abstract methods must be declared abstract.

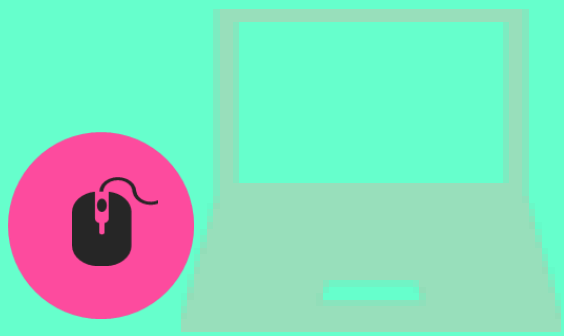
Interface

- An interface is a reference type, similar to a class, that can contain only constants and method signature

```
interface InterfaceName
{
Constant declaration;
Method signature;
}
```

Packages

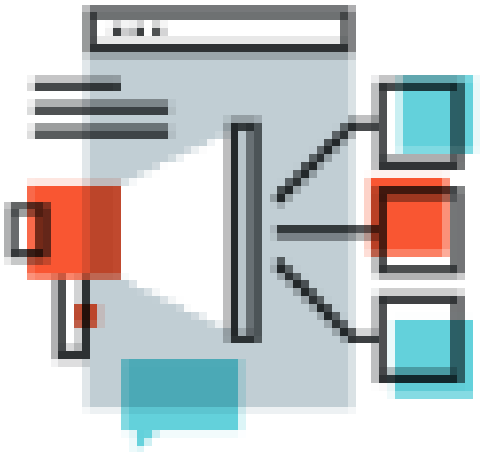
- To locate classes - classes with similar functions can be placed in the same package.



Sample Program - Abstract

```
abstract class Figure {

double dim1;
double dim2;

Figure(double a, double b)
{
dim1 = a;
dim2 = b;
}

abstract double area(); //abstract method
}

class Rectangle extends Figure {

      Rectangle(double a, double b)
      {
      super(a, b);
      }

      double area()
      {
      System.out.println("Inside Area for Rectangle.");
      return dim1 * dim2;
      }
}

 class Triangle extends Figure {
       Triangle(double a, double b)
       {
       super(a, b);
       }

       double area()
       {
       System.out.println("Inside Area for Triangle.");
       return dim1 * dim2 / 2;
       }
 }
 ----------------------------------------------------------

 class AbstractAreas {

       public static void main(String args[]) {

       Figure f = new Figure(10, 10);

       Rectangle r = new Rectangle(9, 5);
       Triangle t = new Triangle(10, 8);

       System.out.println("Area is " + r.area());
       System.out.println("Area is " + t.area());
       }
 }
```

SAMPLE PROGRAM



Interface

```
interface Circle
{
      final static double pi = 3.142;
      double compute(double  x);
}

interface Segi
{
      final static double panjang= 5;
      double kira(int  y);
}

class Shape implements Circle, Segi
{
      public double compute(double  x)
      {
          return(pi * x * x);
      }
      public double kira(int  y)
      {
          return( panjang * y);
      }
}

class Interface3
{
      public static void main(String args[])
      {

Shape sh = new Shape();
System.out.println("Area of Circle = " + sh.compute(10));
System.out.println("Area of Segi = " + sh.kira(3));
      }
}
```

PACKAGE

SAMPLE PROGRAM

```
package mypack;
import info.*;

class cth_pakej{
public static void main(String args[]){
example obj = new example();
obj.cetak();
} }
```

Using packagename.*

```
package info;
public class example{
    public void cetak(){
    System.out.println("JAVA PACKAGE");
    } }
```

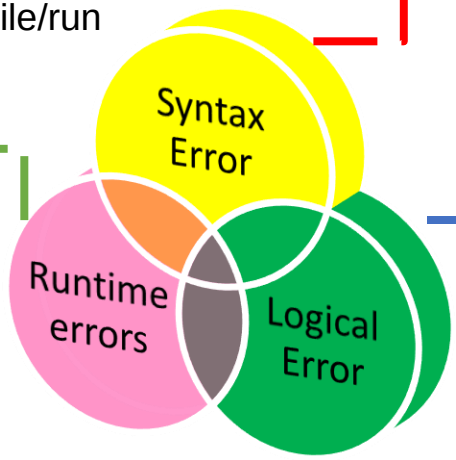
To compile and run program using package

```
javac –d . example.java
javac –d . cth_pakej.java
java mypack.cth_pakej
```

CHAPTER 5

Exception Handling

Error that appears after compile/run a code



Error relating to the programming language used

Error in programming logic

To avoid abnormal program termination.
To avoid system crashes.
To take a necessary action.

EXCEPTION HANDLING

try block → **catch block** → **finally block**

- **try block**: Statements that might cause an exception
- **catch block**: What to do if exception happened
- **finally block**: Statements that have to be execute whether the exception happened or not

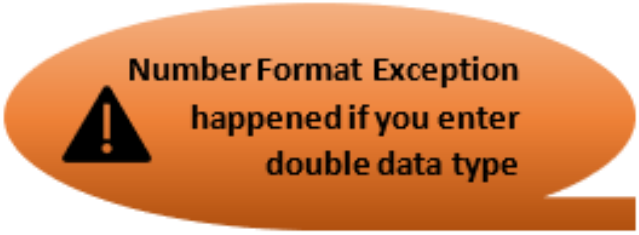
Sample Program
Exception Handling

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

    int var[]={5,10};

    int x = var[2]-var[1];
    System.out.println("Answer: " +x);

    }
```

```
try
    {
    int x = var[2]-var[1];
    System.out.println("Answer: " +x);
    }

catch(ArrayIndexOutOfBoundsException e)
    {
    System.out.println("Array subscript out of range");
    }
```

Sample Program
Exception Handling

```
import java.io.* ;

public class NumExcep
{
      public static void main ( String[] a ) throws IOException
      {
      BufferedReader stdin = new BufferedReader (new
      InputStreamReader(System.in ) );

      String inData;
      int num ;

      System.out.println("Enter an integer:");
      inData = stdin.readLine();

      num   = Integer.parseInt( inData );
      System.out.println("The square of " + inData + " is " + num*num );

   }
}
```



Sample Program
Exception Handling

```
import java.io.* ;

class ArithExcep
{
  public static void main(String args[]) throws IOException
  {
  BufferedReader stdin = new BufferedReader (new
  InputStreamReader(System.in ) );
  String inData;
  int num ;

 System.out.println("Enter an integer:");
 inData = stdin.readLine();
 int a=10, c;

 num    = Integer.parseInt( inData );
 c=a/num;

System.out.println("Answer:" +c);
 }
}
```

! Arithmetic Exception happened if you enter zero

CHAPTER 6

String Manipulation and Multithreading

String is a sequence of characters.

The String class has several methods that can be used to manipulate the string values.

Each method has its own characteristic

Method	Description
length()	To find the number of characters in a string
concat()	To concatenate two strings
charAt()	To returns the character at the specified index from this string
equals()	To compare 2 string are equals or not
toUpperCase()	To return the string in UPPERCASE
toLowerCase()	to return the string in LOWERCASE
replace()	To replace character with another character
substring()	To extracts apart of the string

STRING

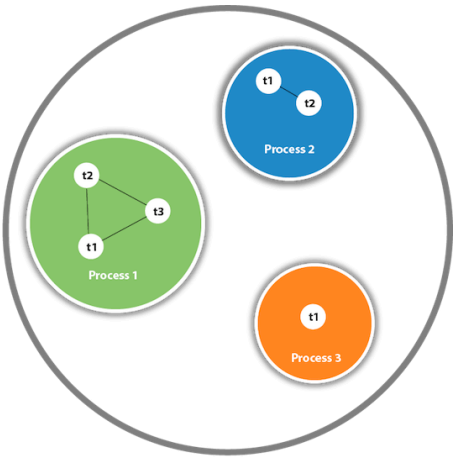
STRING

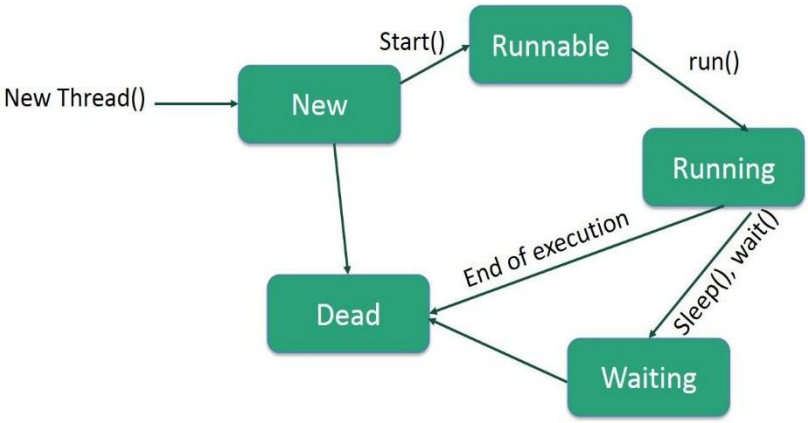
SAMPLE PROGRAM

Example 1

```
public class StringTest2{

        public static void main(String[] args) {

        String message1 = new String("Welcome to JAVA");
        String message2 = new String("Welcome to Oracle Site");
        String message3 = new String("Hewwo");

System.out.println("Length of this string is: "+message1.length());

System.out.println("Concat of this string is:
"+message1.concat(message2));

System.out.println("Character at the index 0 from this string is:
"+message1.charAt(0));

System.out.println("String is equal : "+message1.equals(message2));

System.out.println("Convert to uppercase: "+message1.toUpperCase());

System.out.println("Convert to lowercase: "+message1.toLowerCase());

System.out.println("Replace: "+message3.replace('w', 'l'));

System.out.println("Substring:  "+message1.substring(11,15));
        }
}
```

Thread is executed inside the process. There is context-switching between the threads. There can be **multiple processes inside the OS** and **one process can have multiple threads.**



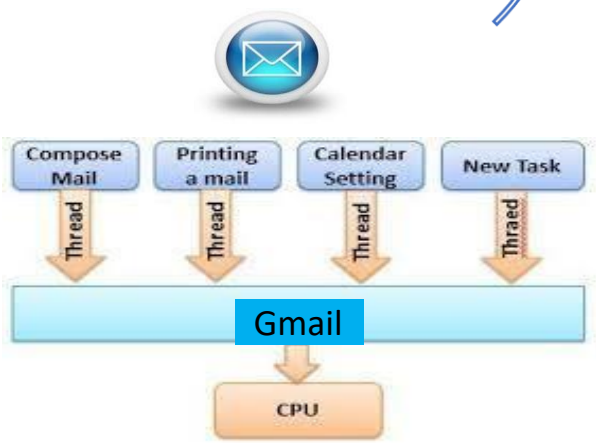


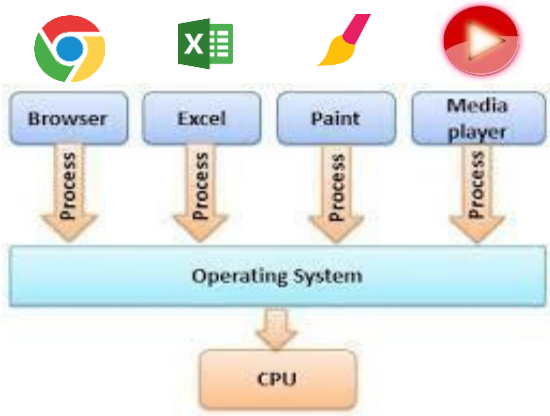
Life Cycle of Thread

MULTITHREADING

Ways to create a thread

- By extending Thread class
- By implementing Runnable interface.





Multithreading VS Multitasking

REFERENCES

Schildt, Herbert. (2022). Java: A Beginner's Guide. (9th ed.). New York: McGraw Hill. (ISBN:978-1260463552)

Herbert. S. (2018). Java: The Complete Reference, Eleventh Edition 11th Edition. United States: McGraw-Hill Education. (ISBN: 978-1260440232)

Horstmann. C. (2018). Core Java Volume I--Fundamentals (Core Series) 11th Edition. New York: Pearson. (ISBN: 978-0135166307)

Juliana Cosmina. (2018). Java for Absolute Beginners: Learn to Program the Fundamentals the Java 9+ Way. United Kingdom: Apress. (ISBN: 978-1484237786)

CODE WITH JAVA –gets you started programming in Java. This book allow you to begin with the basics, how to create, compile, and run a Java program. It also includes example code with commentary that describes the programming techniques.

Highlights -

Fundamentals of Java Programming
Control Structure And Array
Classes, Objects And Methods
Inheritance And Polymorphism
Exception Handling
String Manipulation and Multithreading

e ISBN 978-967-0047-27-0

9 789670 047270