Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

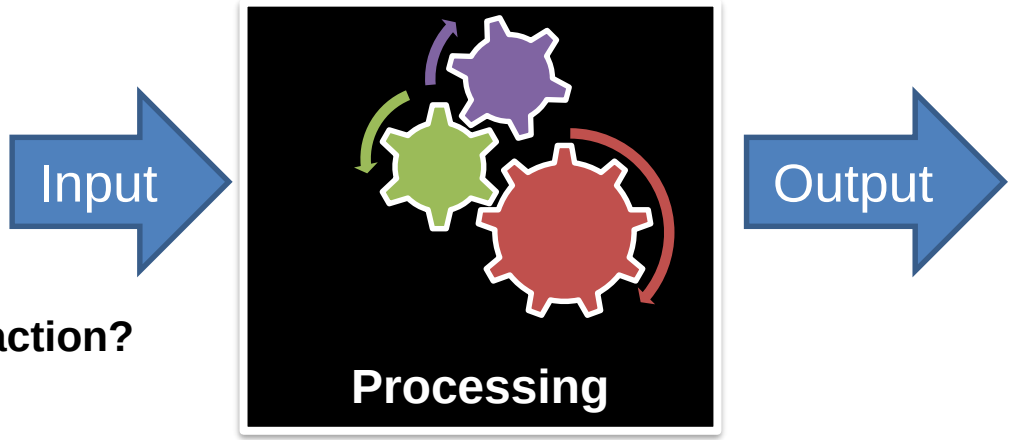
Java Programming For Beginners
Course 1

What is programming ?

What Is Programming and Program Development Life Cycle ?

- Programming is **a process of problem solving**

- **Step 1: Analyze the problem**
  - **Understand the Overall problem**
  - **Define problem requirements**
    - **Does program require user interaction?**
      - **If yes , What Is the Input ?**
    - **What is the Expected output?**
  - **Design steps (algorithm) to solve the problem**



Algorithm:

Step-by-step problem-solving process

What Is Programming and Program Development Life Cycle ?

- **Step 2: Implement the algorithm**
  - **Implement the algorithm in code**
  - **Verify that the algorithm works**

- **Step 3: Maintenance**
  - **Use and modify the program if the problem domain changes**

If the problem is complex, divide it into sub-problems

Analyze each sub-problem as above

Example

- **Write a program to find the Area of a rectangle**

The area of the Rectangle are given by the following formula:

Area = Rect Length * Rect Width.

Input :

Rectangle Length , Rectangle Width.

Processing :

Area = Rect Length * Rect Width.

Output :

Print Out The area.

To sum It up…….

Think

Then

Write Code

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

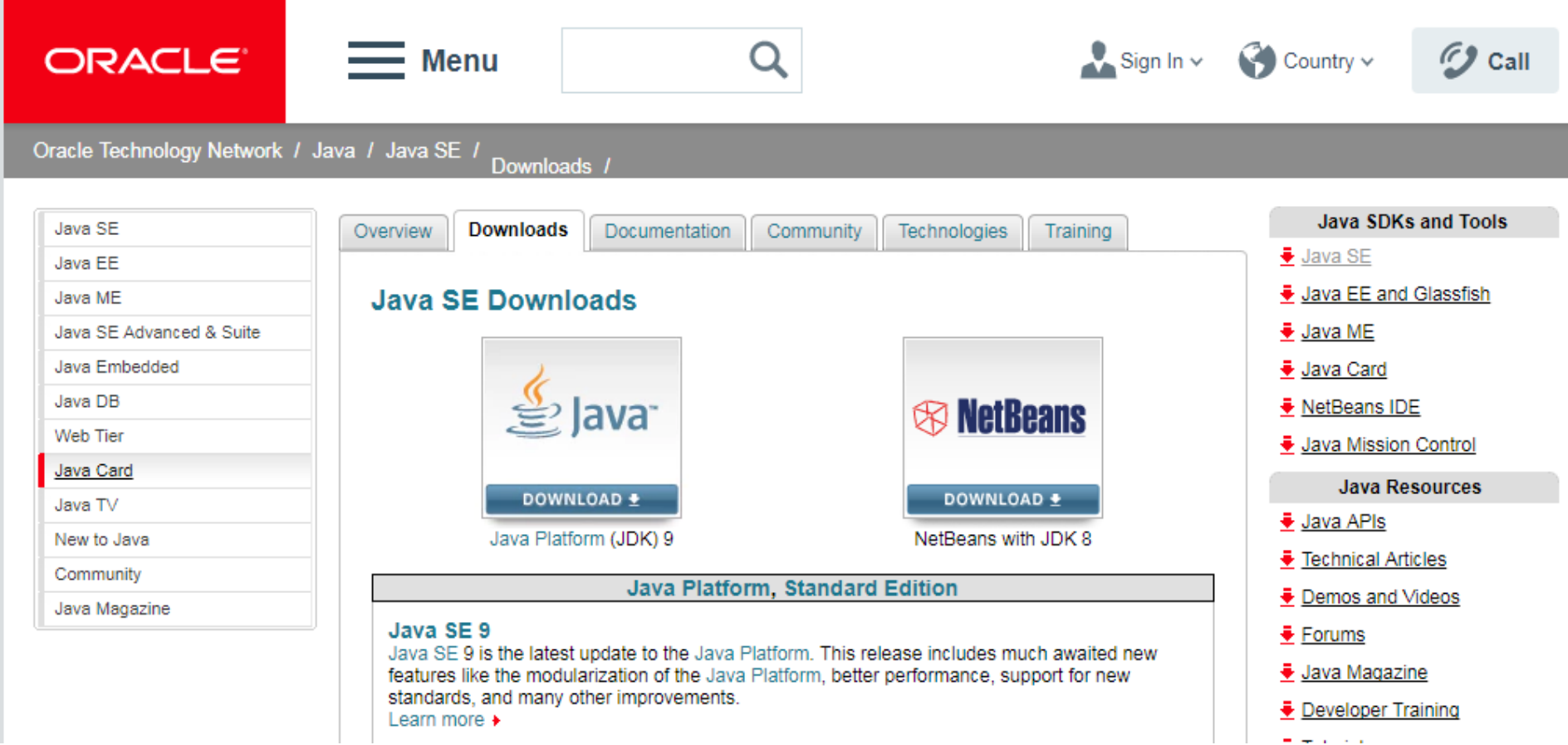
محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1

How to Install Java ?



http://www.oracle.com/technetwork/java/javase/downloads/index.html

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1

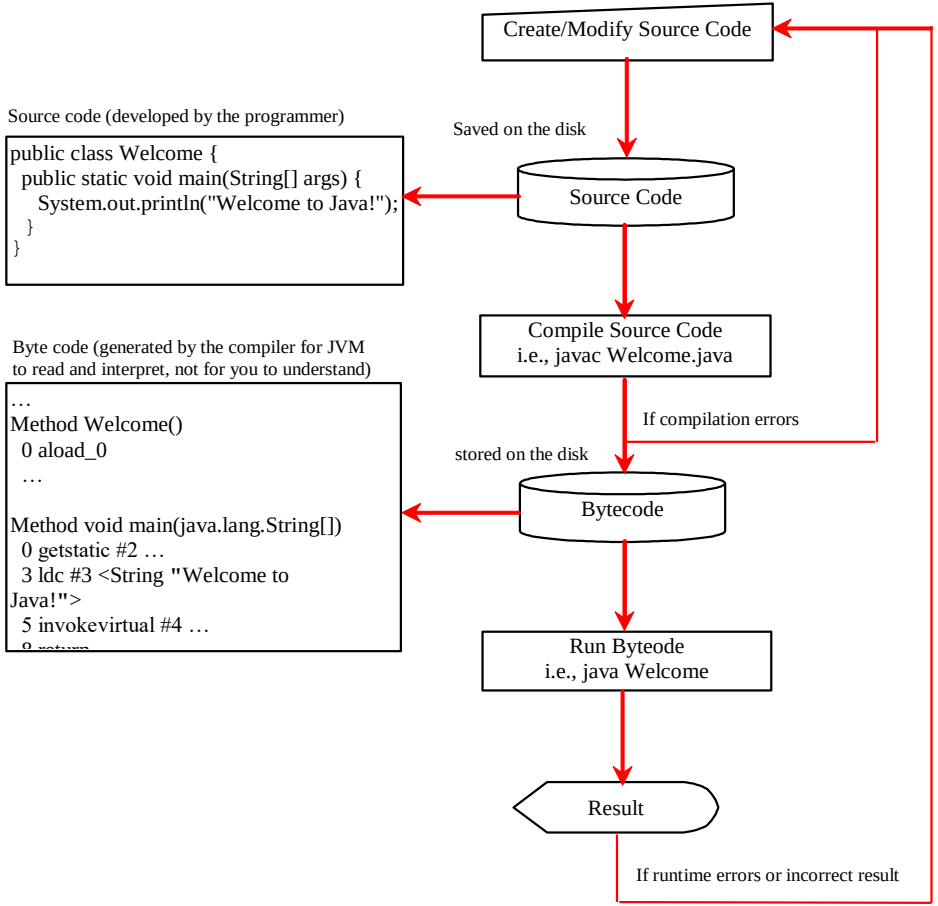
Your First java Program

Your First Java Program

```
1  public class Welcome {
2    public static void main(String[] args) {
3      // Display message Welcome to Java! on the console
4      System.out.println("Welcome to Java!");
5    }
6  }
```

```
Welcome to Java!
```

Processing a Java Program



Programming Errors

- Syntax Errors
  - Detected by the compiler

```
1  public class ShowSyntaxErrors {
2    public static main(String[] args) {
3      System.out.println("Welcome to Java);
4    }
5  }
```

- Runtime Errors
  - Causes the program to abort

```
1  public class ShowRuntimeErrors {
2    public static void main(String[] args) {
3      System.out.println(1 / 0);
4    }
5  }
```

- Logic Errors
  - Produces incorrect result

```
1  public class ShowLogicErrors {
2    public static void main(String[] args) {
3      System.out.println("Celsius 35 is Fahrenheit degree ");
4      System.out.println((9 / 5) * 35 + 32);
5    }
6  }
```

Comments

- We can put comment on our programs using
- **//** to comment a single line

```
// this progrm is written by mohamed El desouki
```

- **/***

  Mulitple Lines

- ***/** to comment multiple lines

```
/* This program is written by Mohamed El Desouki
On Monday 11/1/2018
*/
```

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1

Display Program Output

Interacting With User: Displaying Messages on Screen

- In Java, we use **System.out.print ( ) Or System.out.println ()** To be Display text on The screen

Escape sequence	Description
\n	Newline. Position the screen cursor at the beginning of the *next* line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	Carriage return. Position the screen cursor at the beginning of the *current* line—do *not* advance to the next line. Any characters output after the carriage return *overwrite* the characters previously output on that line.
\\	Backslash. Used to print a backslash character.
\"	Double quote. Used to print a double-quote character. For example, `System.out.println("\"in quotes\"");` displays `"in quotes"`.

Interacting With User: Displaying Messages on Screen

- He said that “Iam From Egypt” but lives in “KSA”
  - System.out.print(“ He Said That \“Iam From Egypt \” but lives in \“KSA\” “)
- The Files located in D:\javaprorg\tryoutput
  - System.out.print(“The Files located in D:\\javaprorg\\tryoutput “)
- Sunday Monday Tuesday Wednesday Thursday
  - System.out.print(“Sunday \t Monday \t Tuesday \t Wednesday\t Thursday“)

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1
Accepting Input From User
Part 2

```java
import java.util.Scanner; // program uses class Scanner

public class Addition {
   // main method begins execution of Java application
   public static void main(String[] args) {
      // create a Scanner to obtain input from the command window
      Scanner input = new Scanner(System.in);

      System.out.print("Enter first integer: "); // prompt
      int number1 = input.nextInt(); // read first number from user

      System.out.print("Enter second integer: "); // prompt
      int number2 = input.nextInt(); // read second number from user

      int sum = number1 + number2; // add numbers, then store total in sum

      System.out.printf("Sum is %d%n", sum); // display sum
   } // end method main
} // end class Addition
```

Java Data Types

char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1byte	true or false
float	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)

Interacting With User: Accept Input From User

- A variable is a location in the computer's memory where a value can be stored for use by a program.
- All variables must be **declared** with a **name** and a **data type** before they can be used in a program.
- Declaration : DataType Identifier

```
int a, b, c;          // Declares three ints, a, b, and c.
int a = 10, b = 10;   // Example of initialization
byte B = 22;          // initializes a byte type variable B.
double pi = 3.14159; // declares and assigns a value of PI.
char a = 'a';         // the char variable a iis initialized with value 'a'
```

Example 1

- **Write a program to find the Area of a rectangle**

The area of the Rectangle are given by the following formula:

Area = Rect Length * Rect Width.

Input :

Rectangle Length , Rectangle Width.

Processing :

Area = Rect Length * Rect Width.

Output :

Print Out The area.

Working With Variable

```
Int  length ;
Int  width;
Int  area;


Scanner input = new Scanner(System.in);

Length =   input.nextInt();

Width =     input.nextInt();

Area =  Length * width ;
```

Example 2

- **Write a program to find the perimeter and area of a square**

The perimeter and area of the square are given by the following formulas:

```
perimeter = Side Length * 4
area = Side Length * Side Length
```

Input:

Square Side Length

Processing:

perimeter = Side Length * 4

area = Side Length * Side Length

Output:

Print Out The Perimeter and Area.

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1

Arithmetic Operators – Part 2

Arithmetic Operations

Name	Meaning	Example	Result
+	Addition	34 + 1	35
-	Subtraction	34.0 - 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder	20 % 3	2

Precedence of arithmetic operations

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. [*Caution:* If you have an expression such as `(a + b) * (c - d)` in which two sets of parentheses are not nested, but appear "on the same level," the C++ Standard does *not* specify the order in which these parenthesized subexpressions will be evaluated.]
*, /, %	Multiplication, Division, Modulus	Evaluated second. If there are several, they're evaluated left to right.
+ -	Addition Subtraction	Evaluated last. If there are several, they're evaluated left to right.

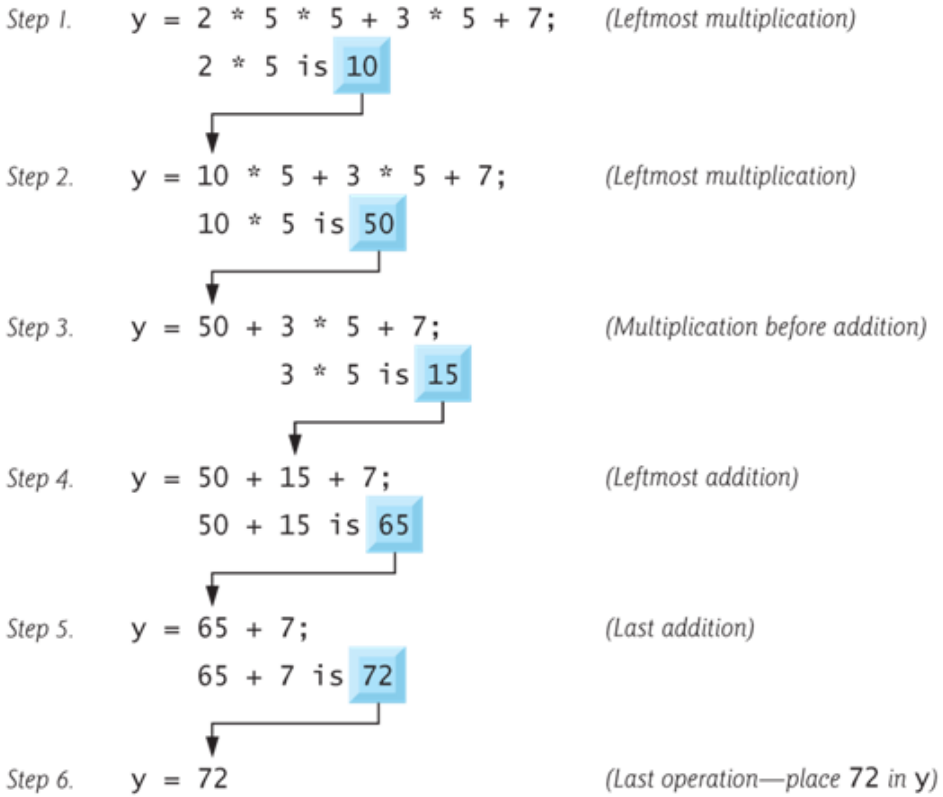
Precedence of arithmetic operators.

For example, `2 + 3 * 5` and `(2 + 3) * 5`

both have different meanings

Precedence of arithmetic operations

Example :



Precedence of arithmetic operations

- ? = **1 + 2 * (3 + 4)**
  - Evaluated as **1 + (2 * (3+4))** and the result is **15**
- ? = **5 * 2 + 9 % 4**
  - Evaluated as **(5*2) + (9 % 4)** and the result is **11**
- ? = **5 * 2 % (7 – 4)**
  - Evaluated as **(5 * 2) % (7 – 4)** and the result is **1**

Data Type of an Arithmetic Expression

- Data type of an expression depends on the type of its operands
  - Data type conversion is done by the compiler

- If operators are *, /, +, or – , then the type of the result will be:
  - integer, if all operands are integer.
    - » Int A , B;
    - » A+ B → Integer.
  - float, If at least one operand is float and there is no double
    - » Int A ; Float B;
    - » A + B → Float.
  - double, if at least one operand is double
    - Int A ; Float B; Double C;
    - » A + B + C → double.

Increment and Decrement Operators

- Increment operator: increment variable by 1
  - Pre-increment: ++variable
  - Post-increment: variable++

- Decrement operator: decrement variable by 1
  - Pre-decrement: --variable
  - Post-decrement: variable --

- Examples :

++K , K++ → k= K+1

--K , K-- → K= K-1

Increment and Decrement Operators

- If the value produced by ++ or – – is not used in an expression, it does not matter whether it is a pre or a post increment (or decrement).

- When ++ (or – –) is used before the variable name, the computer first increments (or decrements) the value of the variable and then uses its new value to evaluate the expression.

- When ++ (or – –) is used after the variable name, the computer uses the current value of the variable to evaluate the expression, and then it increments (or decrements) the value of the variable.

- **There is a difference between the following**

```
x = 5;
Print(++x);
```

```
x = 5;
Print (x++);
```

special assignment statements

- Java has special assignment statements called compound assignments

+= , -= , *= , /= , %=

- Example:

```
X +=5 ;   means   x = x + 5;
x *=10;   means   x = x * 10;
x /=5;    means   x = x / 5;
```

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1

Selection - IF Statement

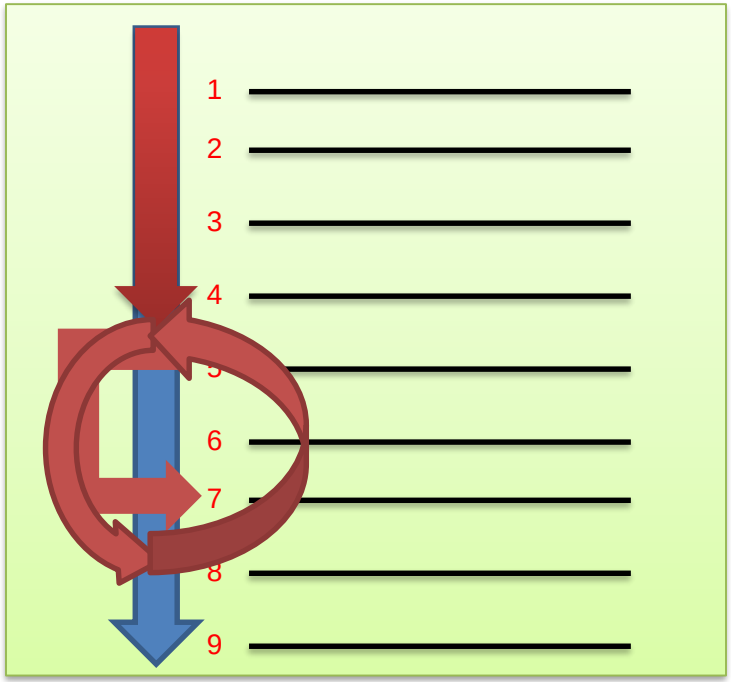
Control Statements

- Normally, statements in a program are executed one after the other in the order in which they're written.
- This is called sequential execution.

- There are control statements enable you to specify that the next statement to be executed may be other than the next one in sequence.
- This is called transfer of control.

- The control statements are categorized in almost two groups:
    - Selection control statements
    - Repetition control statements

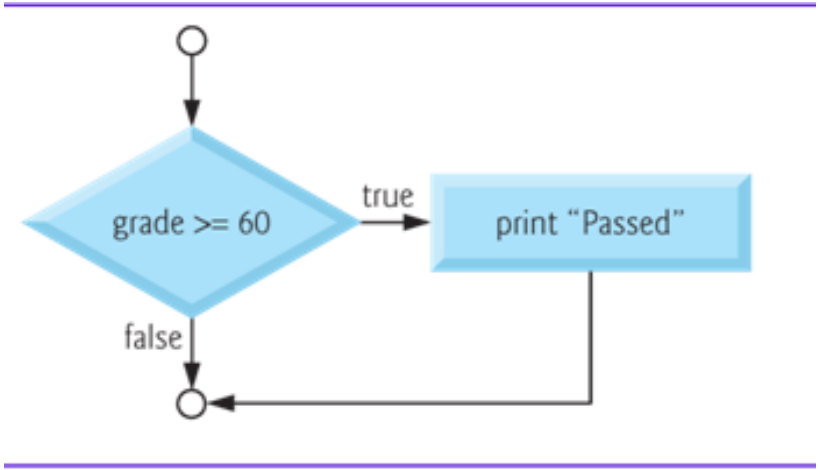
Transfer of Control



Selection Statements : **If Statement**

- Selection statements are used to choose among alternative courses of action.

- For example, suppose the passing mark on an exam is 60. The pseudocode statement
  - If student's marks is greater than or equal to 60 Then
    Print "Passed"



Flowcharting the single-selection `if` statement.

In Java, The syntax for the If statement

```
if ( Expression)
    action statement ;
```

```
if ( Expression)
 {
    action statement 1 ;
    action statement 2 ;
    .
    .
    action statement n ;
}
```

- The Expression can be any valid expression including a relational expression and even arithmetic expression
- In case of using arithmetic expressions , a non-zero value is considered to be true, whereas a 0 is considered to be false

```
if ( grade >= 60 )
System.out.println ("Passed");
```

Relational Expression and Relational Operators

- **Relational expression is an expression which compares 2 operands and returns a TRUE or FALSE answer.**

Example : a >= b , a == c , a >= 99 , ‘A’ > ‘a’

- **Relational expressions are used to test the conditions in selection, and looping statements.**

Operator	Means
==	Equal To
!=	Not Equal To
<	Less Than
<=	Less Than or Equal To
>	Greater Than
>=	Greater Than or Equal To

Selection Statements : **If Statement**

Example : write a program that accept an integer from the user and in case this integer is even print out the following message

“This number is even “ .

Selection Statements : **If .. Else Statement**

- The IF…Else selection statement allows you to specify that there is a course of actions are to be performed when the condition is true and another course of actions will be executed when the condition is false.

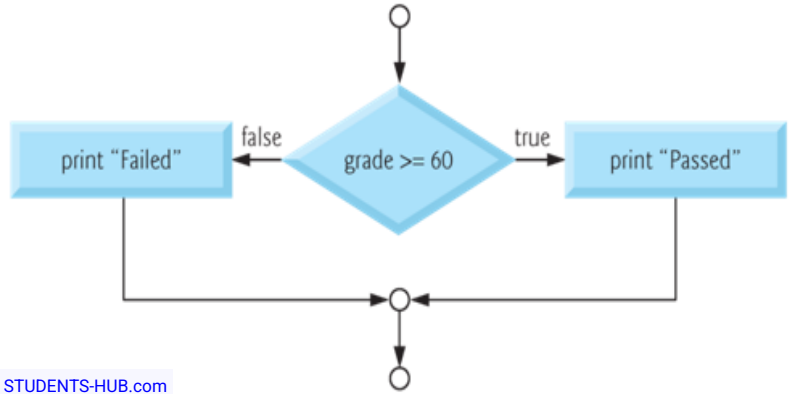
- For example, the pseudocode statement

  – **If** student's mark is greater than or equal to 60

  Print "Passed"

  else

  Print "Failed"

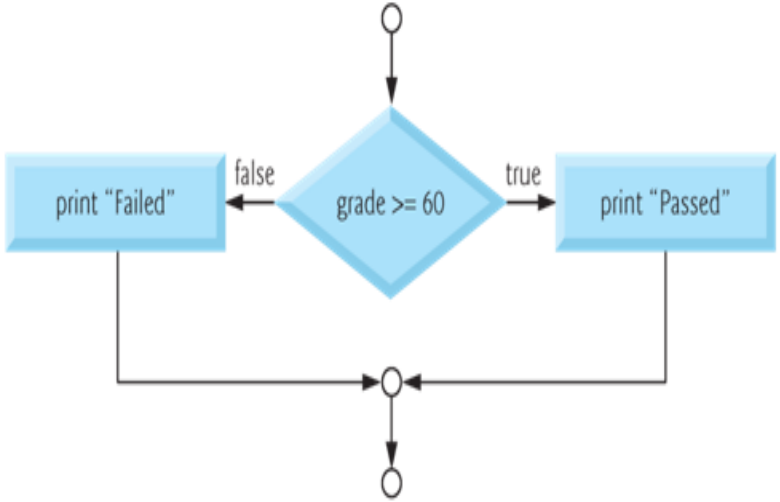


In Java, The syntax for the If…Else statement

```
if ( Expression)
   action statement ;
Else
   action statement ;
```

```
if ( Expression)
 {
   action statements 1 ;
   .
   action statement n ;
}
Else
{
action statements 1 ;
   .
action statement n ;
}
```

```
if ( grade >= 60 )

System.out.println("Passed");
Else
  System.out.println("Failed");
```



Selection Statements : **If – else Statement**

Example : write a program that accept an integer from the user and print out whether it is Positive or Negative number.

Java Programming For Beginners
Course 1

Selection - IF Statement

Nested If

- **Nested If :** **means to write an if statement within another if statement.**

Example : write a program that accept an integer number from the user ,
in case the number is Positive , check and print out whether it is Even or Odd number.

```
int main()
{
int number;
System.out.println("Please Enter any number “) ;
Number = input.nextInt();

if ( number >=0)
        if (number % 2 == 0)
                System.out.println(" This is an Even number “);
        else
                System.out.println("This is an Odd number“);
}
```

IF – Else IF statement

- For example, write a program that ask the user to Enter 2 numbers and print out whether they are equal or there is one which is greater than the other.

```
main()
{
        int num1, num2;
        System.out.println("Enter Number 1 , Number2  “);
        Num1= input.nextInt();

        Num2= input.nextInt();


        if ( num1 == num2 )
                        System.out.println ("Both Are Equal “);
        else if (num1 > num2 )
                       System.out.println("Number 1 is greater than number 2 “);
        else
                        System.out.println ("Number 2 is greater than number 1 “);

}
```

IF – Else IF

- For example, the following code will print
  - A for exam grades greater than or equal to 90,
  - B for grades greater than or equal to 80,
  - C for grades greater than or equal to 70,
  - D for grades greater than or equal to 60, and
  - F for all other grades.

```
– if ( grade >= 90 )
        System.out.println( "A“) ;
  else if ( grade >= 80 )
        System.out.println( "B”);
  else if ( grade >= 70 )
        System.out.println("C”);
  else if ( grade >= 60 )
        System.out.println("D”);
  else
        System.out.println("F“) ;
```

Combining more than one condition

- To combine more than one condition we use the logical operators.

Operator	Means	Description
&&	**And**	The Expression Value Is true If and Only IF both Conditions are true
||	**OR**	The Expression Value Is true If one Condition Is True

Example : check whether num1 is between 0 and 100

```
IF ( num1 >= 0 && num1 <=100 )
    System.out.println ("The Number Is between 1 and 100") ;
Else
     System.out.println( " The Number Is Larger Than 100");
```

Combining more than one condition

Example, print out the student grade according to the following formulas:

- A for exam marks greater than or equal 90 and less than or equal 100 ,
- B for exam marks greater than or equal 80 and less than 90 ,
- C for exam marks than or equal to 70 and less than 80 ,
- D for exam marks than or equal to 60, and less than 70 ,
- F for all other marks.

```
– if ( marks >= 90      && marks <= 100)
      System.out.println ( "A“ );
  else if (marks >= 80 && marks <90 )
    System.out.println ( "B”);
  else if (marks >= 70 && marks <80 )
    System.out.println ( "C”);
  else if (marks >= 60 && marks <70 )
   System.out.println ( "D”);
  else
   System.out.println ( "F\n“) ;
```

Example : A company insures its Employees in the following cases:

- Employee is married.
- Employee is an Single male above 30 years of age.
- Employee is an Single female above 25 years of age.

- **Conditions :**

1. Marital status = ‘M’; **OR**
2. Marital Status =‘S’ and Sex=‘M’ and Age >30 **OR**
3. Marital Status =‘S’ and Sex=‘F’ and Age >25

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1
Selection – Switch Statement
Part 2

Selection statements: Switch Statement.

The switch control statement allows us to make a decision from the number of choices

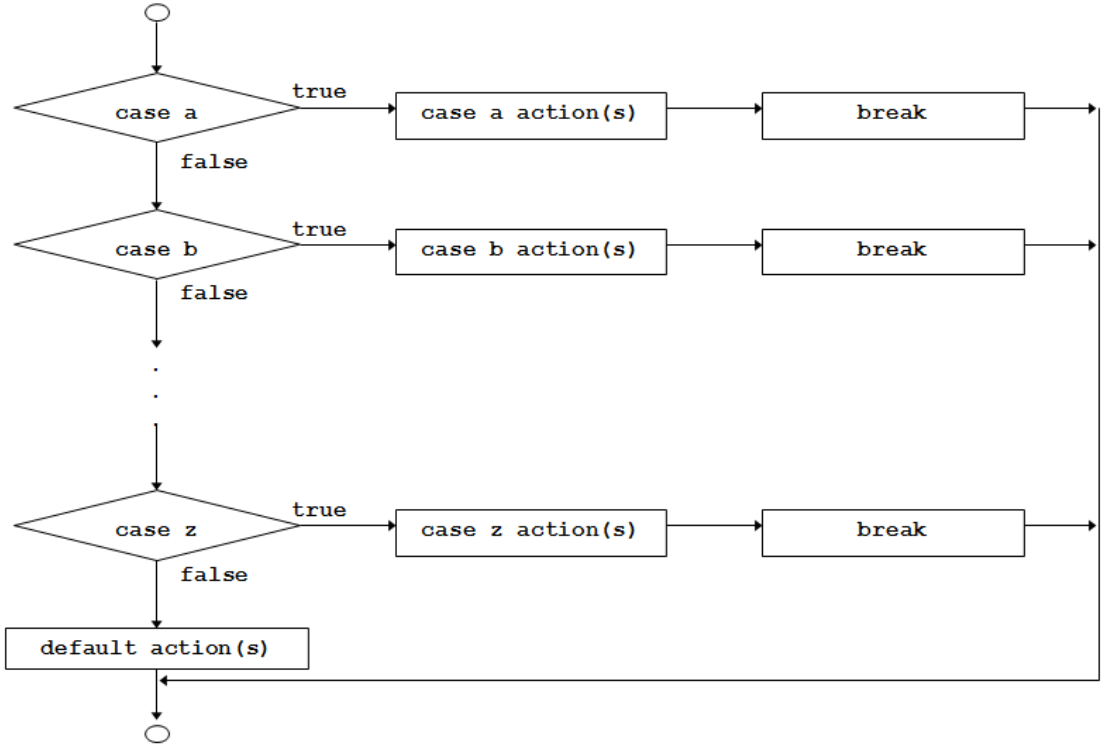
```
Switch (Expression )
{
case constant 1 :
        Action statements;
        break ;
case constant 2 :
        Action statements;
        break ;
case constant 3 :
        Action statements;
        break ;
default :
        Action statements;
}
```

Expression: It could be an integer

```
Switch (10.0);
```

```
Switch (10);

Int i ;
Switch (i);
switch ( 23 + 45 % 4 * k )
```

```
Case  >= 50;
Case   a+ b;
```



1. The switch Statement

```
1  public static void main (String[] args)
2  {
3    //Declaration section
4    Scanner read = new Scanner (System.in);
5    char grade;
6    String message;
7    //input section
8    grade = read.next().charAt(0);
9    //processing section
10   switch (grade)
11   {
12     case 'A': message = "Excellent";
13                         break;
14     case 'B': message = "Very Good";
15                         break;
16     case 'C': message = "Good";
17                         break;
18     case 'D': message = "Fair";
19                         break;
20     case 'F': message = "Failed";
21                         break;
22     default: message ="The grade is invalid";
23   } //end switch
24   //output section
25   System.out.println (message);
26 } //end main
```

```
if (grade == 'A')
   message = "Excellent";
   Else if (grade == 'B')
      message = "Very Good";
    else if (grade == 'C')
         message = "Good";
    else if (grade == 'D')
            message = "Fair";
     else if (grade == 'F')
               message = "Fail";
      else   message = "The grade is invalid";
      System.out.println (message);
```

Write a program that displays the following menu, then acts accordingly:

Enter your choice:
1. Add two numbers
2. Get the Subtract of two number
3. Get the square of a number

The program should give an error message for invalid inputs.

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

    Scanner read = new Scanner (System.in);
   int choice, num1, num2, result = -1;
   String message = “Invalid input”;
   System.out.println (“Enter your choice:”);
   System.out.println (“1. Add two numbers”);
   System.out.println (“2. Get the double of a positive number”);
   System.out.println (“3. Get the square of a number”);
   choice = read.nextInt();
   //processing section
switch (choice)
   {
      case 1: System.out.println (“Enter two positive integers”);  //prompt
                        num1 = read.nextInt();
                        num2 = read.nextInt();
                        result = num1 + num2;  // the value of result is no more equal to  -1
                        break;
 case 2: System.out.println (“Enter a positive integer”);            //prompt
                        num1 = read.nextInt();
                        num2 = read.nextInt();
                         result = num1 – num2;
                         break;
      case 3:  System.out.println (“Enter an integer”);                                  //prompt
                         num1 = read.nextInt();
                         result = num1 * num1; // the value of result is no more equal to  -1
                         break;
default: message =“Invalid value of choice”;   //no change to  “result” ➔ result=-1
   } //end switch
```

- When the same action is to be taken for several case labels, then we may write the program as follows:

```
public static void main (String[] args)
{
   //Declaration section
   Scanner read = new Scanner (System.in);
   char letter;
   String vowel = "This is a vowel";
   //input section
   letter = read.next().charAt(0);
   //processing section
   switch (letter)
   {
      case 'a':
      case 'e':
      case 'o':
      case 'i':
      case 'u': System.out.println (vowel);
                  break;
     default: System.out.println ("This is not a vowel");
    } //end switch
} //end main
```

This is equivalent to:

```
if (letter=='a') || (letter=='e') ||
(letter=='o') || (letter=='i') ||
(letter=='u')
        System.out.println (vowel);
```

switch vs. nested if

- The switch statement is an elegant way to apply multiple selections. It is less confusing.

- However, the programmer is forced sometimes to use the nested if. Examples of such cases include:

  - If the selection involves a range of values.
    - Example: if (score >= 60) && (score < 70)

  - If the selector's type is double or float.
    - Example: if (salary == 5000.25)

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1

Repeation – While Loop

Control Statements : Repetition statements

- Some times we need to repeat a specific course of actions either a specified number of times or until a particular condition is being satisfied.

- For example :
  - To calculate the Average grade for 10 students ,
  - To calculate the bonus for 10 employees or
  - To sum the input numbers from the user as long as he/she enters positive numbers.

- There are three methods by way of which we can repeat a part of a program. They are:
  - **(a) Using a while statement**
  - **(b) Using a do-while statement**
  - (**C) Using a for statement**

Opening Problem

Problem:

```
System.out.println("Welcome to Java!");
...
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
```

```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java");
  count++;
}
```

1- The While Loop

```
While ( continuation condition)
{

 Action statement 1 ;
Action statement  2 ;
.
.
Action statement  n ;
}
```

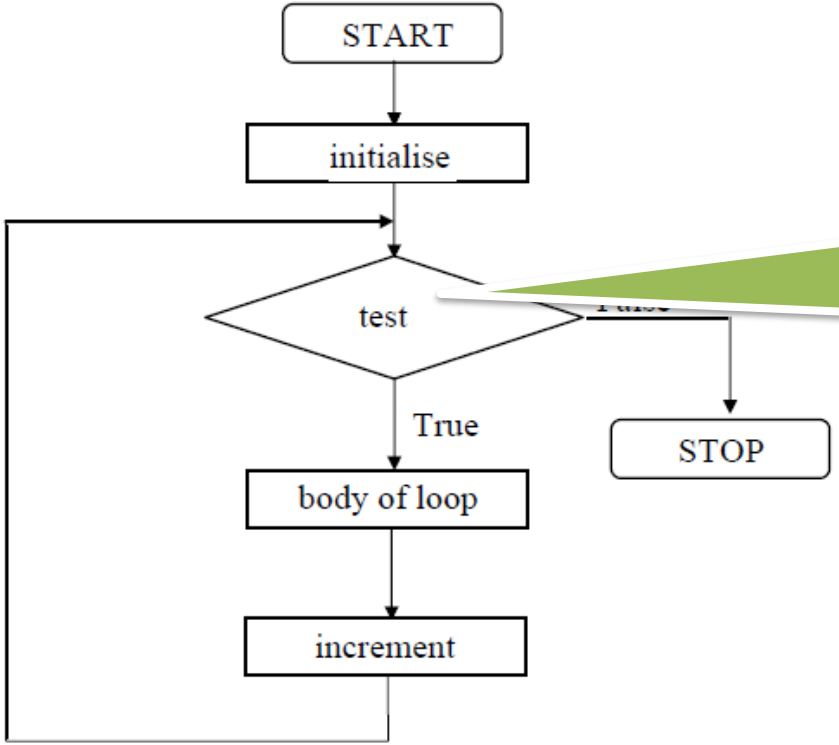
```
initialise loop counter ;
while ( test loop counter using a condition )
{
      do this ;
      and this ;
      increment loop counter ;
}
```

1- loop counter is any numeric variable (int , float ,….).

2- when using a counter , the while loop is called counter–controlled loop.

3- The initial value of the loop counter is up to you,

but you have to increment it to avoid endless loops.



The condition being tested may be relational, logical or even numeric expression :

while (i <= 10)

while (i >= 10 && j <= 15)

While (3) , While (3 + 5)

- **Example** : Write a program that calculates and prints out the Average grade for 6 students .

```
int counter = 1;
int grade=0 , sum=0;
while (counter <=6)
{
System.out.printf(" Enter grade for student no %d" , counter);
grade = input.nextInt( );
sum += grade;
counter ++;
}
System.out.printf(" Average Grade is %f“ , sum/counter );
```

- **Example** : Write a program that calculates and prints out the Average grade for 5 students or ends the program by entering -1.

```
int grade=0, counter =1, sum=0;
System.out.println(" Enter 5 grades or -1 To Exit “);
while (counter <=5 && grade !=-1)
{
grade = input.nextInt( );
sum += grade;
counter ++;
}

System.out.printf(" The sum of grades is %d“,sum);
```

Write a program that randomly generates an integer between 0 and 100, inclusive. The program prompts the user to enter a number continuously until the number matches the randomly generated number. For each user input, the program tells the user whether the input is too low or too high, so the user can choose the next input intelligently. Here is a sample run:

```
// Generate a random number to be guessed
int number = (int)(Math.random() * 101);

Scanner input = new Scanner(System.in);
System.out.println("Guess a magic number between 0 and 100");

int guess = -1;
while (guess != number) {
  // Prompt the user to guess the number
  System.out.print("\nEnter your guess: ");
  guess = input.nextInt();

  if (guess == number)
    System.out.println("Yes, the number is " + number);
  else if (guess > number)
    System.out.println("Your guess is too high");
  else
    System.out.println("Your guess is too low");
} // End of loop
```

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1

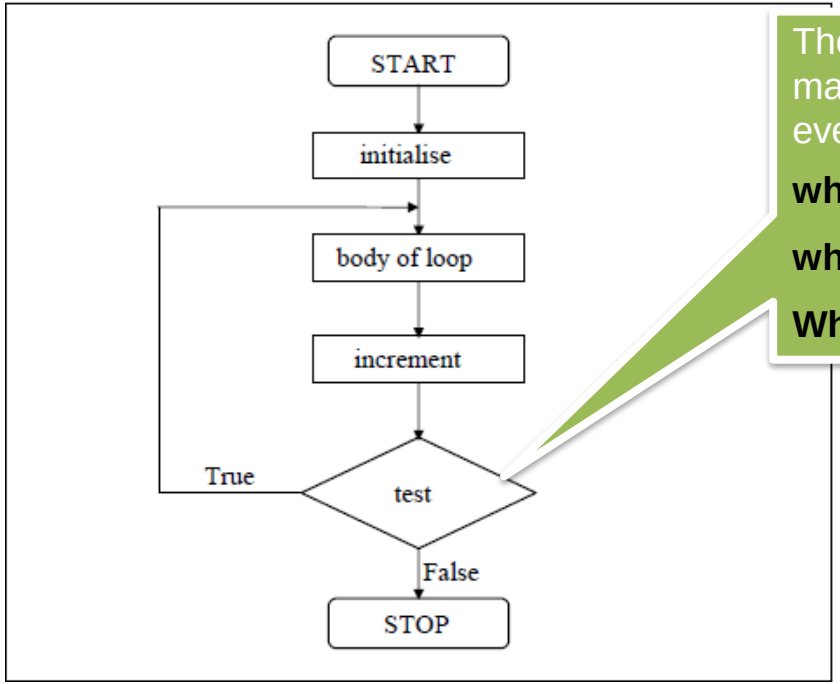
Repeation – do..while Loop

2- The do-while Loop

```
do
{
    this ;
    and this ;
    and this ;
    and this ;
} while ( this condition is true ) ;
```

```
while ( this condition is true )
{
    this ;
    and this ;
    and this ;
    and this ;
}
```

2- The do-while Loop



The condition being tested may be relational, logical or even numeric expression :

while (i <= 10)

while (i >= 10 && j <= 15)

While (3) , While (3 + 5)

2- The do-While Loop

- **Example** : Write a program that calculates and prints out the Average grade for 6 students .

```
int counter = 1;
int grade=0 , sum=0;
do
{
System.out.println("Enter grade for student no “+ counter);
Grade= input.nextInt( );
sum += grade;
counter ++;
}
while (counter <=6) ;

System.out.println(" Average Grade is  “ + sum/counter );
```

```
int option ;
int num1=20 , num2=10;
do {
   System.out.println("Select an option from the MENU");
   System.out.println("1. Sum 2 numbers");
   System.out.println("2. Subtract 2 numbers");
   System.out.println("3. Divide 2 Numbers");
   System.out.println("0. Exit");
   option = input.nextInt();
    switch(option) {

      case 1 :
           System.out.println("sum of the 2 numbers is " + (num1+num2));
           break;
      case 2 :
           System.out.println("subtract of the 2 numbers is " + (num1-num2));
           break;
      case 3 :
          System.out.println("Divide of the 2 numbers is " +( num1/num2));
          break;
     case 0 :
           System.out.println("Good bye");
           break;
      default:
          System.out.println("Invalid Option");
          break;
           }
}while( option !=0 ) ;
```

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1

Repeation – For Loop

3- The For Loop.

- For Loop is probably the most popular looping instruction.

- general form of **for statement is**

```
for ( initialise  counter ; test  counter ; increment  counter )
{
      do this ;
      and this ;
      and this ;
}
```

- The **for allows us to specify three things about** a loop in a single line:

(a) Setting a loop counter to an initial value.

(b) testing the loop counter to detect whether its value reached the number of repetitions desired.

(c) increasing the value of loop counter each time the program segment

within the loop has been executed.

3- The For Loop.

- **Example** : Write a program that calculates and prints out the Average grade for 6 students .

```
int grade=0, sum=0;
for (int counter =1 ; counter <=6 ; counter ++)
{
System.out.println("Enter Grade “);
Grade = input.nextInt();
sum += grade;
}
System.out.println ("The Average grades is “ +( sum/6));
```

```
int counter = 1;
int grade=0 , sum=0;
while (counter <=6)
{
System.out.println("Enter grade”) ;
grade = Input.nextInt( );
sum += grade;
counter ++;
}
System.out.println ("Average Grade is “ + (sum/6 );
```

3- The For Loop.

- **Example** : Write a program that prints out numbers from 0 to 10;

```
for (int i= 0 ; i <=10 ; i++)
System.out.println ( i );
```

- **Example** : Write a program that prints out numbers from 0 to 10 in descending order;

```
for (int i = 10 ; i >=0 ; i-- )
System.out.println ( i);
```

- **Example** : Write a program that prints out the even numbers from 2 to 20;

```
for (int i = 2 ; i <=20 ; i+=2 )
        System.out.println (i);
```

3- The For Loop.

- **Example** : Write a program that accepts 10 numbers from the user and prints out the sum of even numbers and the sum of odd numbers.

```
int i = 1 ;
for ( ; i <= 10 ; i + + )
    System.out.println ( i);
```

```
int i ;
for ( i = 0 ; i++ < 10 ; )
    System.out.println ( i);
```

```
int i = 1 ;
for ( ; i <= 10 ; )
{
    System.out.println ( i);
    i ++;
}
```

```
int i ;
for ( i = 0 ; ++ i < 10 ; )
    System.out.println ( i);
```

3- The For Loop.

- **Example** : Write a program that calculates the Factorial for any given positive number.

  Ex : **Factorial (5) = 5 * 4 * 3 * 2 * 1**

```
int number, factorial=1;
System.out.println ( "Enter a positive number“);
Number = input.nextInt();
if (number < 0 )
System.out.println ( " Enter Positive Numbers only“) ;
else
for (int i= 1 ; i <=number ; i++)
    factorial = factorial * i;
System.out.println ( " Factorial = “ + factorial);
```

Nested Loops

- **Example** : Write a program that calculates the Factorial for numbers from 1 to 10;

```
for ( int number=1; number<=10 ; number++)
{
    for ( int i= 1 ; i <=number ; i++)
    {
        factorial = factorial * i ;
    }
  System.out.println ( " Factorial =  “ + factorial);
}
```

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1
Repeation – Break & continue Statements

THE break STATEMENT

- The break statement is typically used To exit early from a loop (for, while, do…while)

- After the break statement, the remaining of the statements inside the loop are skipped. Then, execution continues starting at the first statement after the loop.

```
1   Int num=0, counter = -1, sum = 0;   //initialize the accumulator sum
2   while (counter <=5)                 //num is used as a sentinel
3      {
4        System.out.println ("Enter a positive integer");   //prompt
5        num = read.nextInt();
6        if (num < 0)
7          {
8            System.out.println ("Negative numbers are not allowed");
9            break;
10         } //end if
11       sum =sum + num;
12       Counter ++;
13     } // end while
   System.out.println ("Sum = " + sum);
```

THE continue STATEMENT

- The continue statement may be used in all loop statements (for, while, do…while)
- The continue statement skips the remaining statements inside the loop; and proceeds with the next iteration, if any.

```
1   Int num=0, counter = -1, sum = 0;   //initialize the accumulator sum
2   while (counter <=5)                 //num is used as a sentinel
3      {
4        System.out.println ("Enter a positive integer");   //prompt
5        num = read.nextInt();
6        if (num < 0)
7          {
8            System.out.println ("Negative numbers are not allowed");
9            continue;
10         } //end if
11       sum =sum + num;
12       Counter ++;
13     } // end while
   System.out.println ("Sum = " + sum);
```

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1

Revision on
Conditional and loop statements

2. ANALYSIS

INPUT & OUTPUT

Saudi Airlines permit each passenger to hold one bag. The maximum weight allowed for the bag depends on the class of the passenger which is: 30 Kg for First class ('F'), 25 Kg for Business ('B') and 20 Kg for Economy ('E'). If the weight of the bag excesses the allowed one, the passenger should pay 10 SR for each Kg.

Write a program that calculates the excess baggage charge for the passenger. All input data should be read by the user. Solve the problem using the **switch** statement.

Charge depends on?

Passenger Class	PassClass (Given)	Allowed Weight (Given)
First Class	'F'	30 Kg
Business Class	'B'	25 Kg
Econonmy Class	'E'	20 Kg

2. ANALYSIS

- INPUT
  - For a specific passenger, we want to know:
    - Passenger's Class ➔ Entered by the user.
    - Baggage Weight ➔ Entered by the user.

- OUTPUT
  - Charge to excess weight:
    - Excess weight = Actual Weight – Allowed Weight *(depending on his class)*
    - Charge = Excess Weight * 10 SR

5. CODE (1)

```
1  // import necessary libraries
2  import java.util.*;
3  public class passenger
4  {
5     // instantiate Scanner object
6     static Scanner read = new Scanner(System.in);
7     public static void main (String[] args)
8      {
9         // Declaration section
10              char passClass;                          // passenger's class
11              double bagWeight;          // passenger's bag weight
12              double excessWeight = 0.0;// passenger's excess weight
13              double charge = -1.0;    // charge on excess weight
14        // Input section
15              System.out.print ("Please enter passenger's class");
16              passClass =  read.next().charAt(0);    // read passenger's class
17              System.out.print ("Please enter passenger's baggage weight");
18              bagWeight =  read.nextDouble();       // read baggage weight
19        // Processing section: charge depends on passClass
20           switch (passClass)
21            {
22              case 'F': excessWeight = bagWeight - 30;
23                        break;
```

5. CODE (2)

```
19          // Processing section: charge depends on passClass
20             switch (passClass)
21              {
22                case 'F': if (bagWeight > 30)
23                                    excessWeight = bagWeight - 30;
24                                 break;
25                case 'B': if (bagWeight > 25)
26                                 excessWeight = bagWeight - 25;
27                              break;
28                case 'E': if (bagWeight > 20)
29                                    excessWeight = bagWeight - 20;
30                              break;
31              } //end switch
32             charge = excessWeight * 10;
33          // Output Section
               System.out.printf ("Charge = %5f", charge);
     } //end main
} //end class
```

5. CODE (3)

WITH USER INPUT VALIDATION

```
19          // Processing section: charge depends on passClass
20             switch (passClass)
21              {
22                case 'f':
23                case 'F': if (bagWeight > 30)          {excessWeight = bagWeight -
24 30;
25                                                              charge =
26 excessWeight * 10;}
27                                    break;
28                case 'b':
29                case 'B': if (bagWeight > 25)          {excessWeight = bagWeight -
30 25;
31                                                              charge =
32 excessWeight * 10;}
33                                     break;
34                case 'e':
35                case 'E': if (bagWeight > 20)          {excessWeight = bagWeight -
36 20;
37                                                       charge = excessWeight * 10;}
38                           break;
39                 default: System.out.println ("Invalid passenger class");
40               } //end switch
          // Output Section
```

See you Next Video

Java Programming For Beginners
Course 1

محمد إبراهيم الدسوقى

محاضر بكلية هندسة و علوم الحاسب – قسم نظم المعلومات

جامعة الأمير سطام بن عبد العزيز - السعودية – محافظة الخرج

Email : mohamed_eldesouki@hotmail.com

Java Programming For Beginners
Course 1

Functions (Methods) – Part 1

Functions - Methods

- Most computer programs that solve real-world problems include hundreds and even thousands of lines.
- Experience has shown that the best way to develop and maintain a large program is to construct it from smaller pieces or modules, each of which is more manageable than the original program.
- **A Function :** is a self-contained block of statements that perform a specific task.
- Using a **function** is something like hiring a person to do a specific job for you.

In Real Life

In Programming

Task 1

Function ( )

Task 2

Function ( )

Task 3

Function ( )

Task 4

Function ( )

Why to use Functions ?

- **Functions** allows to write more manageable units of code.
- Writing functions avoids rewriting the same code over and over.
- Easy maintenance for programs.

Important Tips

1- Don’t try to cram the entire logic in one function. It is a very bad style of programming.

2- Instead, break a program into small units and write functions for each of these isolated subdivisions.

3- Don’t hesitate to write functions that are called only once.

Calculate and print out The sum and the Average of 3 student marks.

Task 1

Get_Marks( )

Task 2

Calc_sum( )

Task 3

Calc_Average ( )

Task 4

Print_out ( )

Functions

Function Definition

```
return-value-type function-name( parameter-list )
{
Lines of code to be
executed

...
(Body)
}
```

return-value- Type:
Is the data type for the returned value from the function to the calling program. It is **Mandatory, if no returned value expected , we use keyword Void.**

Parameter-List : Is a list of data values supplied from the calling program as input to the function. It is **Optional**

2- Invoking Function (From The main( ))

Function name (*Actual Parameter- List*);

Function Types

1. Built in Functions (Ready Made).

- For Example : Math Functions Like

int abs (int number) ;

double pow (double base, double exponent) ;

Double floor (double number);

Double sqrt (double number);

Java Programming For Beginners
Course 1

Functions (Methods) – Part 2

User-Defined Functions

- Value-returning functions: have a return type
  - Return a value of a specific data type using the `return` statement
  - the returned value can be used in one of the following scenarios:
    - **Save the value for further calculation**

      Int result;

      Result = sum (x , y);

    - **Use the value in some calculation**

      Result = sum(x , y) /2;

    - **Print the value**

      System.out.println (sum(x , y)) ;

- Void functions: do not have a return type
  - *Do not* use a `return` statement to return a value

```
public class methodOneParameter
  {
    public static void main(String[] args)
    {
      System.out.println (“Start of Program”); //output line #1
      drawLine();     //method calling
     Sysetm.out.println(“Welcome to the First lecture in Functions”);
    drawLine();     //method calling
      System.out.println (“End of Program”);
    } //end of main
   //method definition: This method draws a line
   public static void drawLine( )
   {
     int i;
     for (i=0; i < 10; i++)
       System.out.print (‘*’);
       System.out.println();
    } //end of drawLine
} //end of class
```

```
public class methodOneParameter
  {
    public static void main(String[] args)
    {
      System.out.println (“Start of Program”); //output line #1
      drawLine(‘*’);     //method calling
      Sysetm.out.println(“Welcome to the First lecture in Functions”);
      drawLine(‘#’);     //method calling
      System.out.println (“End of Program”);
    } //end of main
   //method definition: This method draws a line
   public static void drawLine( char ch)
   {
     int i;
     for (i=0; i < 10; i++)
       System.out.print (ch);
       System.out.println();
    } //end of drawLine
} //end of class
```

```
public class methodOneParameter
  {
    public static void main(String[] args)
    {
      System.out.println (“Start of Program”); //output line #1
      drawLine(‘*’,15);     //method calling
     Sysetm.out.println(“Welcome to the First lecture in Functions”);
    drawLine(‘#’,20);     //method calling
     System.out.println (“End of Program”);
    } //end of main
   //method definition: This method draws a line
   public static void drawLine(char ch , integer length )
   {
     int i;
     for (i=0; i < length; i++)
       System.out.print (ch);
       System.out.println();
    } //end of drawLine
} //end of class
```

See you Next Video

Java Programming For Beginners
Course 1

Functions (Methods) – Part 3

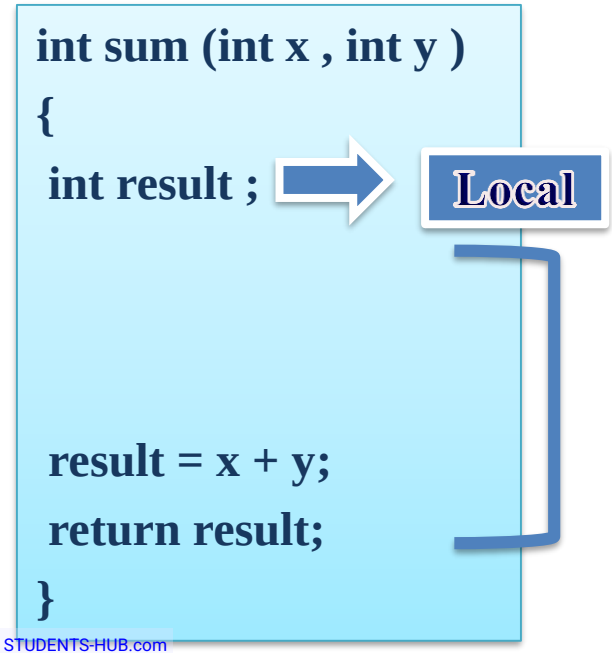
For Example : write a program that ask the user to Enter 3 integer numbers and print out their sum and Average.

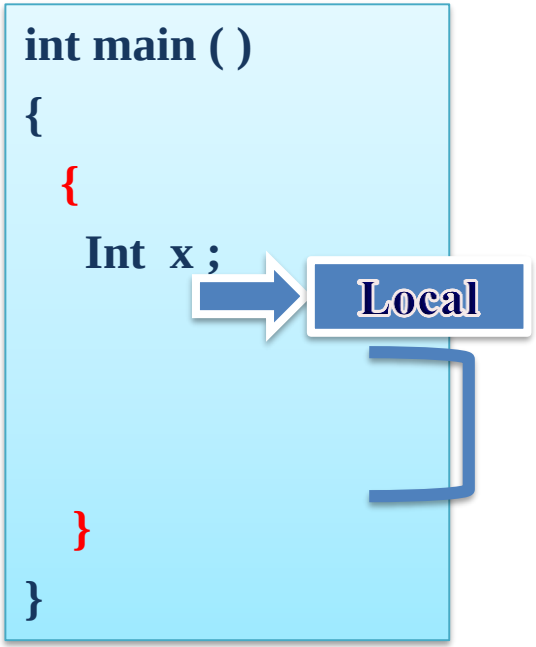
```
int main ()
{
int n1 , n2 , n3 ;
System.out.println("Please Enter 3 integer numbers “ );
n1= input.nextInt();
n2= input.nextInt();
n3= input.nextInt();
 System.out.println(" The sum of the 3 number is “ || sum (n1, n2,n3) );
System.out.println(" The average of the 3 number is " || average (n1, n2,n3) );
}
Public static int sum (int num1 , int num2, int num3)
{
return num1+num2+num3;
}

Public static double average (int num1, int num2, int num3 )
{
return sum (num1 , num2 , num3)/3 ;
}
```

Scope of a variable

scope is the context within a program in which a variable is valid and can be used.

- ***Local variable***: declared within a function (or block)
- can be accessible only within the function or from declaration to the end of the block.
- Within nested blocks if no variable with same name exists.

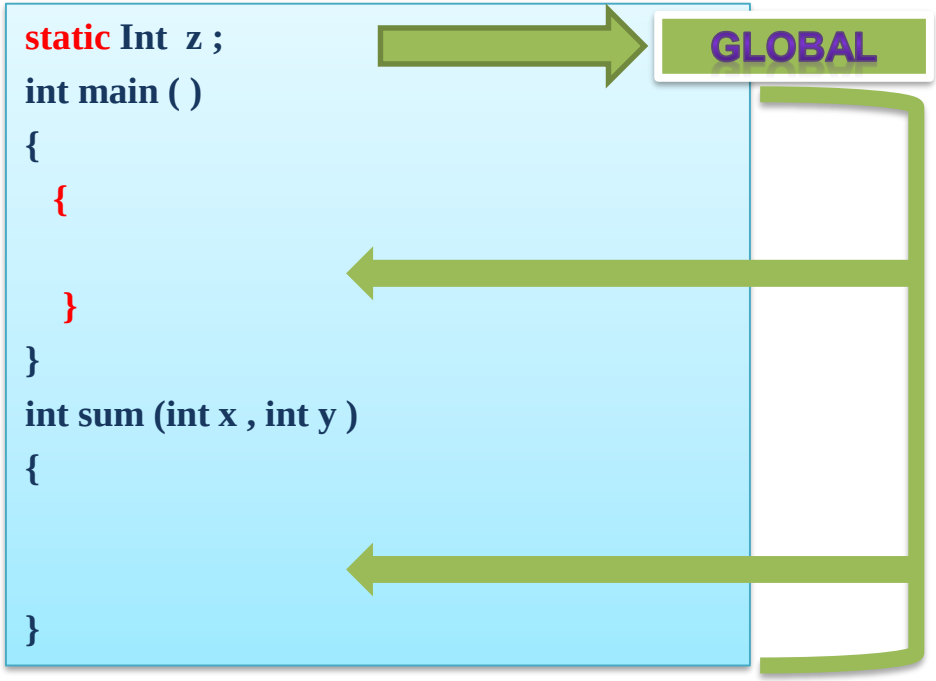




Scope of a variable

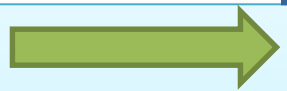
Global variable: declared outside of every function definition.

– Can be accessed from any function that has no local variables with the same name. In case the function has a local variable with the same name as the global variable ,



Scope of a variable

```
Static int x = 100 ;
int main ( )
{
int x= 10;

      {
            int  z , x ;
            z=100;
            y=100;
            x= 250;
            System.out.println(" inner block " << x ;
      }
}
```



GLOBAL

Local

Local

See you Next Video

Java Programming For Beginners
Course 1

Functions (Methods) – Part 4

Method Overloading

- **Method Overloading:**

  Defining several methods within a class with the same name. As long as every Method has a different signature

- **Method Signature:**

  The signature of a method consists of the following:
  - Method name
  - Formal parameter list

Note that the method type is not part of its signature.

- **Examples:**
  - public int Sum(int x, int y)
  - public int Sum(int x, int y , int z)
  - public double Sum(double x, double y, double z)
  - public int Sum(int x, double y)
  - public int Sum(double y, int x)

- The following methods are incorrectly overloaded; the compiler generates an error:

- Example (1): The following methods are incorrectly overloaded because they have the same method name and same formal parameter lists:

  - public void methodABC (int x, double y)
  - public int methodABC (int x, double y)

- Example (2): Changing the names of the formal parameters, does not allow overloading of the previous counter-example:

  - public void methodABC (int x, double y)
  - public int methodABC (int num1, double num2)

- Counter-example (3): Adding the modifier static does not allow overloading of the previous example:

  - public static void methodABC (int x, double y)
  - public int methodABC (int num1, double num2)

Note that the method type and modifiers are not part of the overloading rules

See you Next Video

ARRAY DECLARATION

SYNTAX

dataType[ ] arrayName **=** new **datatype [intExp];**

- In Java, an array is an object that must be instantiated using the new operator.
- arrayName is the reference variable.
- **dataType** is the data type of the array; which is the same type of all elements in the array.
- **intExp** is any expression that evaluates to a positive int type.
- **intExp** represents the size of the array: this is the number of elements in the array.

ARRAY DECLARATION

EXAMPLES

- Consider the following array declaration:

```
int[] num = new int [5];
```

- From the above statement:
  - num is the array name
  - num consists of 5 elements
  - The type of the elements is int

Address	Value
num →	0
	0
	0
	0
	0

- The memory layout will be as follows:
  - As shown, the array name points to the first element in the array.
  - The five elements of the array are adjacent in the memory.

ARRAY DECLARATION

USING CONSTANTS AS SIZES

```
final int ARRAY_SIZE = 5;
int[] num = new int [ARRAY_SIZE];
double[ ] num = new double[ARRAY_SIZE];
char[ ] x = new char[ARRAY_SIZE];
boolean[ ] flags = new boolean[ARRAY_SIZE];
```

- **The size of the array must be of type int.**

3. ARRAY DECLARATION

3.3 USING EXPRESSIONS AS SIZES

```
1 final int ARRAY_SIZE = 5;
2 int[] num = new int[ARRAY_SIZE*5];
```

- has the same effect as:

```
1 int[] num = new int[25];
```

- The expression may be in terms of an int variable rather than a constant:

```
1 int i = 8;
2 int[] num = new int[i*2-1];
```

- has the same effect as:

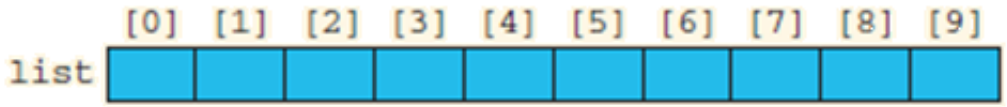
```
1 int[] num = new int[15];
```

Arrays

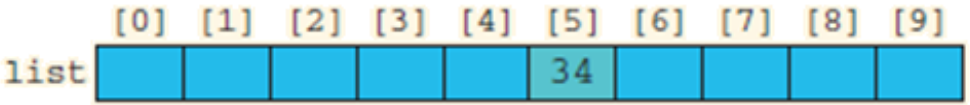
- Accessing array Elements

Arrayname [element index].

- Example: `int list[10];`



`list[5] = 34;`



`System.out.println( list [5];`

Array Initialization

- Consider the declaration

```
int list[10];  //array of size 10
```

- After declaring the array you can use the For .. Loop to initialize it with values submmited by the user.

- Using ***for*** loops to access array elements:

```
for (int i = 0; i < 10; i++)
   //process list[i]
```

- Example:

```
for (int i = 0; i < 10; i++)
     cin >> list[i];
```

- **Example** : **Write a program that ask the user to enter 10 Employee salaries and store them , then add a bonus of 10 % for each employee and print out the average salary value.**

Array Initialization

- Arrays can be initialized during declaration
  - In this case, it is not necessary to specify the size of the array
    - Size determined by the number of initial values in the braces

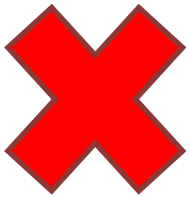
- Example 1: `int Items[] = {12, 32, 16, 23, 45};`

- Example 2: `int items[10] = { 0 };`

- Example 3: `int items[10] = { 5, 7, 10 };`

Array Initialization

```
int Arr1[5];

int Arr2 [5];

Arr1 = Arr2 ;
```



Array as a Parameter to Function

- Arrays are passed by reference only
- The symbol & is *not* used when declaring an array as a formal parameter
- The size of the array is usually passed as a parameter to the function.

```
Float calc_Average ( Float marks [ ] , int size )
{
float sum =0 ;

for (int I =0 ; I <size ; I ++)
    Sum += marks [ I ]  ;

 return sum / size;
}
```

- We may add keyword **const** before array name to Prevent the function from modifying the array elements.

```
Float calc_Average ( const Float marks [ ] , int size )
```

- **Example** : **Write a program that uses a function to search for an item within an array.**

```
bool find_Item( int list[ ] , int searcheditem , int size)
{
int  indx;
bool  found = false;

for ( indx = 0 ; indx < size ; indx++)
    if (list[indx] == searcheditem)
    {
      found =true ;
      break ;
    }
       return found;
}
```

Two Dimensional Array

- Used when data is provided in a table form.
- For Example , to store 4 Marks for 6 students.

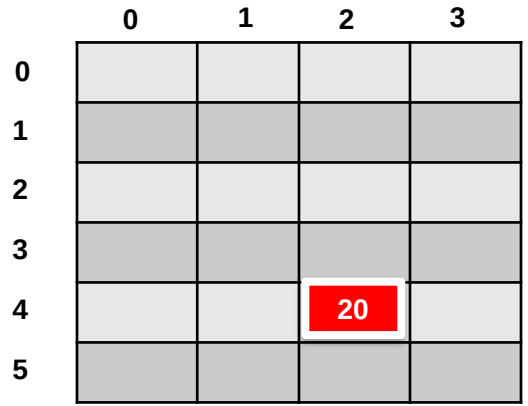
	M 1	M2	M3	M4
Student 1				
Student 2				
Student 3				
Student 4				
Student 5				
Student 6				

- **Two dimensional Array declaration**

```
Datatype ArrayName [ Rows] [Columns] ;
```

Example :

```
Float  marks [6] [4] ;
 marks [4][2]= 20 ;
```



Two dimensional Array Initialization

- Consider the declaration

```
float marks[6][4];
```

- After declaring the array you can use the For .. Loop to initialize it with values submmited by the user.

- Using 2 nested **for** loops to access array elements:

```
for (int row = 0; row < 6; row++)
     for (int col = 0; col < 4; col++)
    cin >> marks[ row ][col];
```

Two dimensional Array Initialization

- Two dimensional Arrays can be initialized during declaration

- Example 1:

```
float marks[4][3] = {{20,30,35},
                     {40,45,65},
                     {60,65,75},
                     {80,65,45}};
```

- **Example** : **Write a program that build a matrix of 5 rows and 3 columns . As the use to enter the values for all the matrix items , print out the sum of all matrix items and print out the sum of the diagonal items.**

Two dimensional Array as a Parameter to Function

when declaring a two-dimensional array as a formal parameter, you can omit the size of the first dimension, but not the second;

that is, you must specify the number of columns.

```
void printMatrix( int  matrix[  ][4],  int  Rows)
  {
     int row, col;
     for (row = 0; row < Rows ; row++)
      {
      for (col = 0; col < 4; col++)
       System.out.println( setw(5) << matrix [row]
  [col] ;
      }
     System.out.println("\n" ;
  }
```