# Software Design and Architecture



## Lab # 07

Loops, Arrays, and Functions in Java.

## Instructor: Mazhar Iqbal Email: [mazhar.iqbal@nu.edu.pk](mailto:mazhar.iqbal@nu.edu.pk) Course Code: SL2002 Semester Spring 2023

Department of Computer Science,

## National University of Computer and Emerging Sciences FAST Peshawar Campus

Table of Contents

[Loops 3](#_bookmark0)

1. [for loop 3](#_bookmark1)
2. [while loop 4](#_bookmark2)

[while loop Example 4](#_bookmark3)

[do while loop 4](#_bookmark4)

[do while loop Example 5](#_bookmark5)

[Break Statement 5](#_bookmark6)

[Continue Statement 6](#_bookmark7)

[Java Random Class 6](#_bookmark8)

[Java Random Class Methods 6](#_bookmark9)

[Java Random Example 7](#_bookmark10)

[Random Numbers 7](#_bookmark11)

[Game of Random Numbers 7](#_bookmark12)

[Array 8](#_bookmark13)

[Syntax of one-dimensional array in C++ 8](#_bookmark14)

[Initialization of Array 9](#_bookmark15)

[Enhanced for loop (for-each loop) 11](#_bookmark16)

[Two dimensional arrays 12](#_bookmark17)

[Declaration of 2D Array 13](#_bookmark18)

[Java Function/Methods 17](#_bookmark19)

[Functions Types 17](#_bookmark20)

* 1. [Built in Functions or standard library methods 17](#_bookmark21)
  2. [User Defined Functions 18](#_bookmark22)

[Function/Method Overloading 20](#_bookmark23)

# Loops

* In computer programming, a **loop** is a sequence of instructions that is continually repeated until a certain condition is reached.
* A **loop** statement allows us to execute a statement or group of statements multiple times

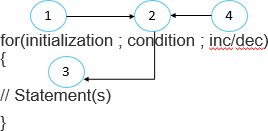
#### Types of Loop

1. for loop
2. while loop
3. do while loop
4. for-each loop (Enhanced For Loop)

## for loop

* for loop is used to a statement or group of statement for a fixed number of times.
* If the number of iterations is fixed then it is recommended to use for loop.

#### Syntax:



for loop Example

class ForDemo

{

public static void main(String args[])

{

int i;

for(i=0; i<=10 ; i++)

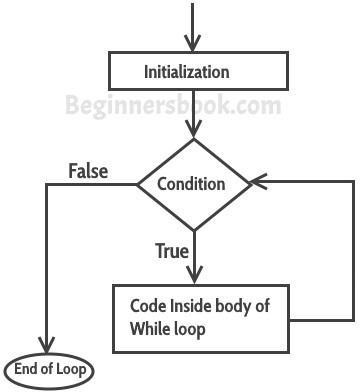
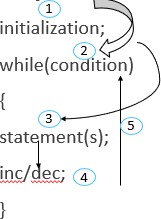
{ System.out.println(“Kmayab Jawan Program”); }

}

}

## while loop

Is used when number of iterations is not fixed.



#### Syntax

while loop Example

class WhileDemo

{

public static void main(String args[]) { int i=1;

while(i<=10)

{

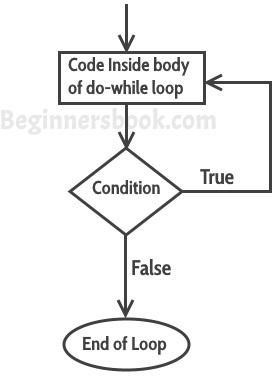
System.out.println(“Kmayab Jawan Program”); i++;

}

} }

## do while loop

* An indefinite loop. Best used when the number of iterations is unknown.



**Syntax** initialization; do{ statement(s); inc/dec;

}

while(condition) ;

do while loop Example

class DoWhileLoopExample {

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

do

{

System.out.println(i);

i-- ; //decrementation

}

while(i>1) ;

}

}

## Break Statement

The break statement terminates the execution of the loop when it is used inside the body of the loop.

**Syntax:** break;

## Continue Statement

* It continues the current flow of the program and skips the remaining code at specified condition.

**Syntax:** continue;

### import java.util.Scanner; Class AddNoWhile

{ // class body starts

### public static void main(String args[])

{ // main method body starts Scanner input = new Scanner(System.in) int n1, n2;

### String choice = “Yes”; // choice declaration and initialization

while(choice.equals(“Yes”))

### { // while body starts

System.out.println(“Enter 1st no”); n1=input.nextInt(); System.out.println(“Enter 2nd no”); n2=input.nextInt(); System.out.println(“Sum=”+(n1+n2));

### System.out.println(“Do you want to do more addition (Yes/No)”); choice= input.next();

} // while loop body closed

### System.out.println(“Thank you for using while loop”);

} // main method body closed

} // class body closed

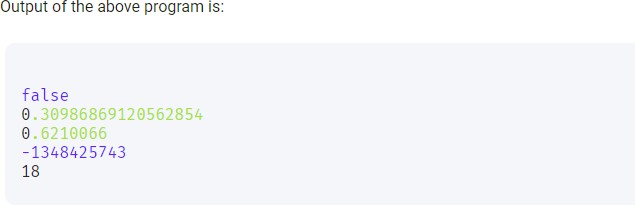
# Java Random Class

* Random class is part of java.util package.
* An instance of java Random class is used to generate random numbers.
* This class provides several methods to generate random numbers of type integer, double, long, float etc.

## Java Random Class Methods

1. nextBoolean(): This method returns next pseudorandom which is a boolean value from random number generator sequence.
2. nextDouble(): This method returns next pseudorandom which is double value between 0.0 and 1.0.
3. nextFloat(): This method returns next pseudorandom which is float value between 0.0 and 1.0.
4. nextInt(): This method returns next int value from random number generator sequence.
5. nextInt(int n): This method return a pseudorandom which is int value between 0 and specified value from random number generator sequence.

## Java Random Example



## Random Numbers

import java.util.Random; Random rand = new Random(); int num = rand.nextInt(6)+1;

Here 6 means it will generate numbers from 0 to 5 means it will generate 6 numbers.

## Game of Random Numbers

import java.util.Random; import java.util.Scanner; public class Game

{ // Class body starts

Public static void main(String args[])

{ // main method body starts int num, guess, count=0;

Scanner input = new Scanner(System.in); Random rand = new Random();

num = rand.nextInt(100)+1;

System.out.println(“Number Generated, try to guess it”); while(true)

{

count++;

System.out.println(“Enter your guess: ”); guess = input.nextInt();

if(guess>num)

{

System.out.println(“Your number is high, please try again”);

}

else if(guess<num)

{

System.out.println(“Your number is low, please try again”);

}

else

{

break;

}

} // while loop body closed

System.out.println(“You found the guess in” + count+ “Attempts”); if(count<5)

{

System.out.println(“Excellent”);

}

if(count>5)

{

System.out.println(“Good”);

}

} // Main method body closed

} // class body closed

# Array

Same name which store multiple values. It is a collection of similar type of elements that have contiguous memory location.

#### Array is:

* 1. Linear data structure (consecutive location)
  2. Static data structure (fixed size)
  3. Homogeneous data will be stored.

## Syntax of one-dimensional array in C++

Datatype arrayName[size];

#### Example:

int array[5];

Datatype arrayName[] = new Datatype[size];

#### Example:

int marks[] = new int[5];

OR

int[] marks = new int[5];

OR

int marks[]; // declaration

marks = new int[5]; //initialization

## Initialization of Array

int marks[] = new int[5] // 5 is array length or size marks[0] = 80 ; // [0] is array index and 80 is array element marks[1] = 90 ;

marks[2] = 70 ; marks[3] = 60 ; marks[4] = 30 ;

# Declaration, instantiation and initialization of java array

int marks[] = {80, 90, 70, 60, 30};

# Declare array length constant

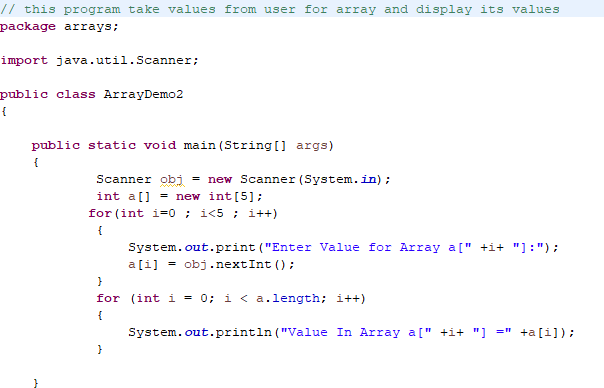
#### Example

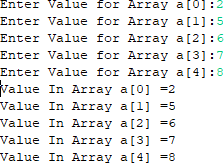
final int ARRAY\_LENGTH = 10 ;

int[] array = new int[ARRAY\_LENGTH];

# Program 1

**Program 2 Runtime value in array from user**





## Enhanced for loop (for-each loop)

* Works with array.
* Is used for traversing in array.
* It is easy to use than simple for loop because we do not need to increment or decrement counter variable.

#### Syntax

for (data type variable : arrayName)

{ statement(s); }

#### Data type must be same as that of array data type.

* Start from 1st element.
* End in last element.
* We cannot use it in reverse order.
* We cannot traverse element in middle of array.
* Only one step incrementation is possible.

# Enhanced for loop example 1

**class** ForEachExample1{

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

//declaring an array

**int** arr[]={12,13,14,44};

//traversing the array with for-each loop

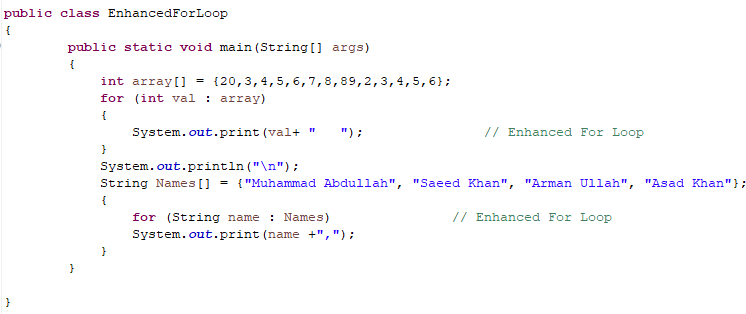
**for**(**int** i:arr){

System.out.println(i+ “ ”);

} }

}

# Enhanced for loop example 2



# Two dimensional arrays

* An array that is represented with two indices/subscripts is called 2D array.
* It is similar to matrix in maths.
* Logically it consists of rows and columns.
* 2D array is called an array of an arrays.

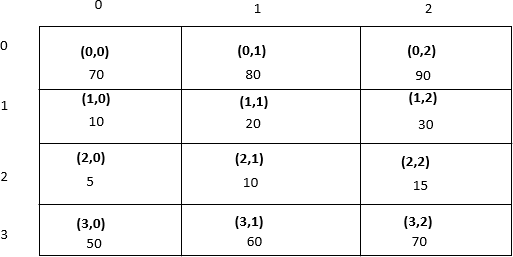
#### Syntax of declaration

Datatype arrayName[][] = new datatype[R][C];

// R means number of rows and C means number of columns

**Example:** int StudentMarks[][] = new int[4][3];

# 2D Array logical Representation



## Declaration of 2D Array

Datatype [][] arrayRefVar;

OR

Datatype [][]arrayRefVar;

OR

Datatype arrayRefVar[][];

OR

* Datatype []arrayRefVar[];

# Instantiation of 2D Array

int[][] array = new int[3][3];

**1st Method**

int studentMarks[][] = new int[4][3]; studentName[0][0] = 70;

studentName[0][1] = 90;

studentName[0][2] = 90;

studentName[1][0] = 10;

studentName[1][1] = 20;

studentName[1][2] = 30;

studentName[2][0] = 5;

studentName[2][1] = 10;

studentName[2][2] = 15;

studentName[3][0] = 50;

studentName[3][1] = 60;

studentName[3][2] = 70;

**2nd Method**

int studentMarks[][] = { { 70, 80, 90 },

{ 10, 20, 30 },

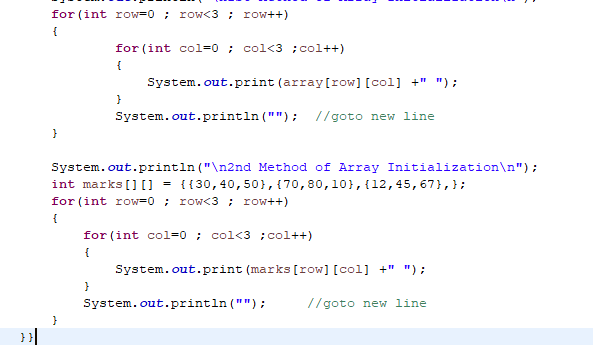
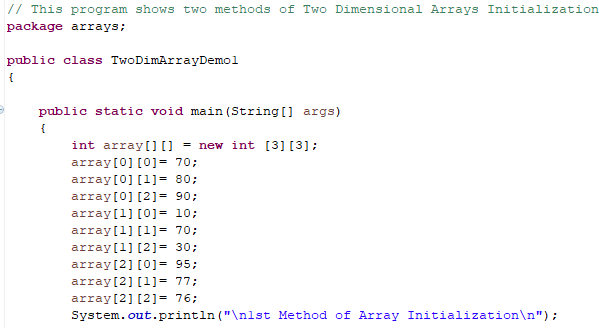
{ 5, 10, 15 },

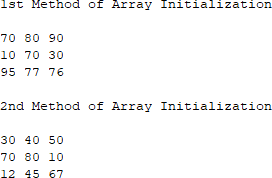
{ 50, 60, 70 },

}; OR

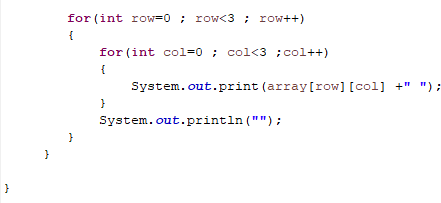
int studentMarks[][] = { { 70, 80, 90 }, { 10, 20, 30 }, { 5, 10, 15 }, { 50, 60, 70 } ,};

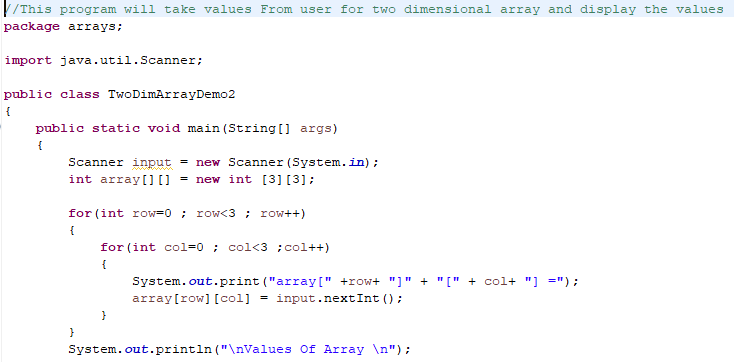
# 2D Array Program 1





**2D Array Program 2**





# Java Function/Methods

* Function is a set of instructions that are designed to perform a specific task.
* A function is a complete and independent program.
* It is executed by the main method to perform its tasks.
* Functions are used to write the code of a large program by dividing it into smaller independent units.
* It avoids the replication of code in the program.

# Functions VS Methods

**Function** — a set of instructions that perform a task.

**Method** — a set of instructions that are associated with an object.

#### METHODS

A method, like a function, is a set of instructions that perform a task. The difference is that a method is associated with an object, while a function is not.

## Functions Types

1. Built in Functions or standard library methods

The standard library methods are built-in methods in Java that are readily available for use.

#### Example:

* 1. println()
  2. nextInt() III.showMessageDialog IV.showInputDialog etc.

1. User Defined Functions

We can also create methods of our own choice to perform some task. Such methods are called user- defined methods.

#### Example:

public static void myMethod() { System.out.println("My Function called");

}

# Function definition

Access specifier return type methodName(list of parameter)

{

statement(s);

}

**The function definition is called method header.**

# Calling Method or Invoking Method

* Executing the statement(s) of method to perform task is called calling a function.
* Calling a method is called invoking a method.

#### Example:

addition();

1. Function have no parameters list and return type

Class Test

{

public void printStar()

{

System.out.println(“\*\*\*\*\*”);

}

public static void main(String args[])

{

Test object=new Test();

object.printStar(); //method call

}

} // Test Class body closed

1. Function have no return type but parameter list

Class Test1

{

public void sum(int x, int y) // formal arguments

{

int sum=x+y;

System.out.println(“Result is”+sum);

}

public static void main(String args[])

{

Test1 object=new Test1(); object.sum(5,6); //Actual Arguments

}

} // Test1 class body closed

1. Function return values

Function can return only one value.

**Return Statement:** The return statement is used to return calculated value from function definition to calling function.

**Syntax:** return x; Class Test2

{

public int sum(int x, int y) // formal arguments

{

return(x+y);

}

public static void main(String args[])

{

Test2 object=new Test2();

int result = object.sum(5,6); //Actual Arguments

System.out.println(“Result is”+result);

}

} // Test2 Class body closed

## Function/Method Overloading

* Method having same name with different set of parameters (type, order, number) then such kind of method is called overloaded method and this mechanism is called method overloading.
* Method overloading is compile time polymorphism or static binding.
* It increases the readability of the program.

**Note:** In java method overloading is not possible by changing the return type of method. Class methodOverloading

{

public void sum(int x, int y) // formal arguments

{

System.out.println(“sum of int is”+(x+y));

}

public void sum(double x, double y) // formal arguments

{

System.out.println(“sum of double is”+(x+y));

}

public void sum(int x, double y) // formal arguments

{

System.out.println(“sum of int & double is”+(x+y));

}

public void sum(double y, int x) // formal arguments

{

System.out.println(“sum of double & int is”+(x+y));

}

public static void main(String args[])

{

methodOverloading object=new methodOverloading(); object.sum(3,5);

object.sum(3.3,5.6);

object.sum(3,5.4);

object.sum(3.6,5);

}

}