Module-1 (Part-I): Introduction to OOP using Java

by:

Dr. Soumya Priyadarsini Panda

Sr. Assistant Professor

Dept. of CSE

Silicon Institute of Technology, Bhubaneswar

Programming Languages

- A programming language is a notation designed to give instructions to a machine or a computer to perform specific tasks.
    - Examples: C, C++, Java, Python, etc

- Two broad categories of programming languages:
    - **Procedural programming languages**
    - **Object Oriented programming languages**

Procedural Vs. Object Oriented Programming

Procedural Programming Language	Object Oriented Programming Language
Specifies a series of well-structured steps and procedures with its programming context to compose a program.	Based on the concept of "objects", which can contain data (attributes/ properties) and code (procedures/ methods).
A program written in a procedural language contains one or more procedures.	Programmers define the data type of a data structure as well as the types of operations (functions) that can be applied to the data structure.
Examples: C, FORTRAN, ALGOL, COBOL, BASIC, etc.	Examples: Java, C++, C#, Python, etc.

Object Oriented Programming (OOP)

- OOP refers to a type of computer programming where-
  - programmers defines the data type of a data structure along with the types of operations (functions) that can be applied to the data structure.
- The data structure becomes an object that includes both data and functions

Features of OOP

- **Object**
- **Class**
- **Abstraction**
- **Encapsulation**
- **Polymorphism**
- **Inheritance**

Object

- Object means a real word entity.
- It can be physical and logical.
- Any entity that has state and behavior is known as an object.
- **Examples:**
  - chair, pen, table, keyboard, bike etc.

Class

- Class is the logical construct which defines the shape and nature of an object.

- It is a collection of similar objects.

- Contain data and methods bundled together under a unit.

- ***Object is an instance of a class***

- Example:
  - If ‘student’ is a class, any student say ‘Amit’ belonging to that class is an instance of the class called an object.

Abstraction

- Humans manage complexity through abstraction.

- For example, people do not think of a car as a set of different individual parts. They think of it as a well-defined object with its own unique behaviour in abstract form as a car.

- Hiding internal details and showing functionality is known as abstraction.

Encapsulation

- Encapsulation is the mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse.

- One way to think about encapsulation is as a protective wrapper that prevents the code and data from being arbitrarily accessed by other code defined outside the wrapper.

- Access to the code and data inside the wrapper is tightly controlled through a well-defined interface.

- A java class is the example of encapsulation.

Polymorphism

- Polymorphism (from Greek, meaning “many forms”) is a feature that allows one interface to be used for a general class of actions.
- The specific action is determined by the exact nature of the situation.

- Consider a stack that uses LIFO mechanism-
  - A program may requires 3 types of stacks (int, float, char).
  - In a non–object-oriented language, 3 different sets of stack routines are required to be created, with each set using different names. However, the algorithm that implements each stack is the same.

  - Because of polymorphism in Java, a general set of stack routines can be defined that all share the same names.

- In java, ***method overloading*** and ***method overriding*** is used to achieve polymorphism.

Inheritance

- Inheritance is the concept of inheriting or deriving properties of an existing class to another class.

- There may be some common features or characteristics that may be needed by number of classes.
  - Those features can be placed in a common class called ***base class*** and the other classes can inherit the properties and define only the new features for their classes.
  - The class that inherits the properties of a base class is called a ***derived class*** or sub class.

- **Advantages:** It helps in reducing the code size since the common characteristics are placed separately in the base class and are just referred in the derived class. This ensures ***code reusability***.

Advantages of OOP

- OOPs provide ability to simulate real-world event much more effectively.

- **Improved software-development productivity:**
  - Object-oriented programming is modular, extensible, and can also be reused therefore provides improved software-development productivity over traditional procedure-based programming techniques.

- **Improved software maintainability:**
  - Object oriented software is easier to maintain. Since the design is modular.

Cont…

- **Faster development:**
  - Reuse enables faster development.
- **Lower cost of development:**
  - The reuse of software also lowers the cost of development.
- **Higher-quality software:**
  - Faster development of software and lower cost of development allows more time and resources to be used in the verification of the software.
- **Data Protection or security**:
  - Achieved by the concept of data hiding.

Disadvantages of OOP

- The thought process involved in object-oriented programming may not be natural for some people, and it can take time to get used to it.

- **Larger program size:**
  - Object-oriented programs typically involve more lines of code than procedural programs.

- **Slower programs:**
  - Object-oriented programs are typically slower than procedure based programs, as they typically require more instructions to be executed.

- **Not suitable for all types of problems:**
  - There are problems that lend themselves well to procedure-based programming style instead of OOP.

Introduction to Java Programming

- Java is an Object oriented programming language developed by **James Gosling** at **Sun Microsystems** and released in 1995

Where Java is used ?

- Desktop Applications
  - Acrobat reader, Media player, Antivirus etc.
- Web Applications
  - irctc.co.in, javatpoint.com etc.
- Enterprise Applications
  - banking applications
- Mobile
- Games etc.

Why Java ?

- Java is the best programming language in terms of opportunities, development and community support.

Major Reasons:

- **Java is Easy to learn:**
  - Java has fluent English like syntax with minimum magic characters

- **Java is an Object Oriented Programming Language:**
  - Developing OOP application is much easier, and it also helps to keep system modular, flexible and extensible.

- **Java has rich API:**
  - A list of prewritten classes provide a tremendous amount of functionality to a programmer
- **Java is Free**

Cont…

- **Powerful development tools**
  - Coding in Integrated Development Environment (IDE): Eclipse and Netbeans provides powerful debugging capability which is essential for real-world development.
- **Wonderful Community Support**
- **Great collection of Open Source Libraries:** Open source libraries makes Java development easy, faster and cost-effective.

- **Java is Platform Independent:**
  - The idea of platform independence is great, and Java's tagline ***"write once run anywhere (WORA)"*** was enticing enough to attract lots of new development in Java.

- **Java is everywhere:** It's on the desktop, mobile, almost everywhere and so is Java programmers.

Features of Java/ The Java Buzzwords

- **Simple**
- **Secured**
- **Portable**
- **Object-Oriented**
- **Robust**
- **Multithreaded**
- **Architectural Neutral**
- **Interpreted**
- **High Performance**
- **Distributed**
- **Dynamic**

Features of Java/ The Java Buzzwords

Simple:

- Java is very easy to learn and its syntax is simple, clean and easy to understand.
- Removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc

Secured:

- Java is best known for its security. With Java, we can develop virus-free systems.

Portable:

- Java is portable because it facilitates to carry the java byte code to any platform

Cont…

Object-Oriented:

- Java is based an Object-oriented programming, therefore software is organized as a combination of different types of objects that incorporates both data and behaviour. It simplifies software development and maintenance.

Robust:

- Java is robust because it uses strong memory management to allocate and de-allocate memory.
- There is automatic garbage collection in java.
- Lack of pointers in java avoids security problem.
- There is object-oriented exception handling and type checking mechanism in java.

Cont…

Multithreaded:

- A thread is like a separate program, executing concurrently.
- Java supports multithreaded programming, which allows writing programs that do many things simultaneously.

Architectural Neutral:

- There is no implementation dependent features e.g. size of primitive types is fixed.
- In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture.
- But in java, it occupies 4 bytes of memory for both 32 and 64 bit architectures.

Cont…

Interpreted and High Performance

- Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java **bytecode.**

- This code can be executed on any system that implements the Java Virtual Machine.

- Java bytecode is carefully designed so that it would be easy to translate directly into native machine code for high performance without losing any benefits of the platform-independent code.

- Java is faster than traditional interpretation since bytecode is "close" to native code.

Cont…

Distributed

- It facilitates to create distributed applications.
- It is designed for the distributed environment of the Internet.

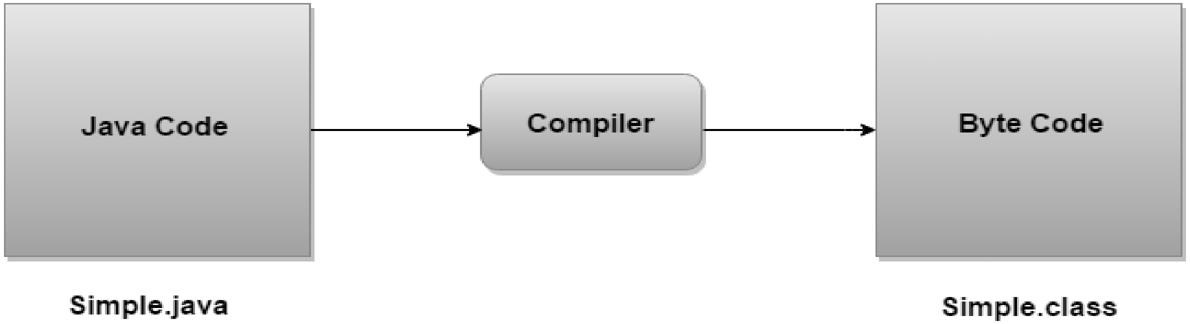
Dynamic

- Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time.

Overview of Java Programming

The Java Byte Code

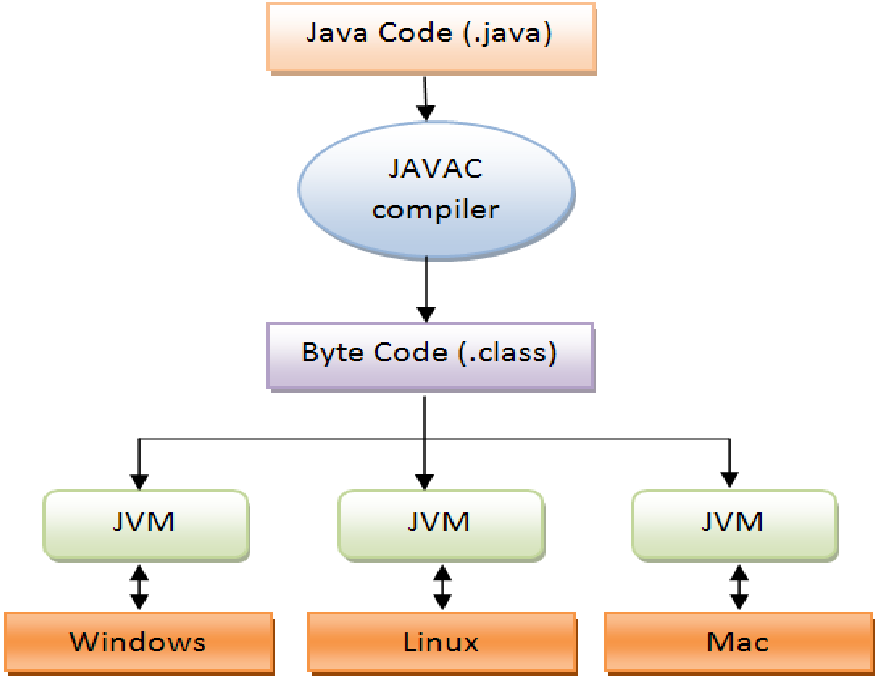
- At compile time, java file is compiled by Java Compiler and converts the java code into **bytecode**



Java's Magic - The Bytecode

- The output of a Java compiler is the ***bytecode*** instead of executable code

- Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system which is called **Java Virtual Machine (JVM)**.



What is JVM?

- Java Virtual Machine (JVM) is an engine that provides runtime environment to drive the Java Code or applications.
- It converts Java bytecode into machines language.
- JVM is a part of Java Run Environment (JRE)

Why Java is Platform Independent?

- When the Java program runs in a particular machine it is sent to java compiler, which converts this code into intermediate code called bytecode.

- This bytecode is sent to JVM which resides in the RAM of any operating system.

- JVM recognizes the platform it is on and converts the bytecodes into native machine code.

Requirements for Writing and Running a Java Program

- Install JDK
- Setting path of the jdk/bin directory
- Use any text editor: Notepad/ Notepad++ to write the code

Compiling and Running java Program

*/*If file name: simple */*

Save as: simple.java

Compile: javac simple.java

Run: java simple

Example: Writing First Java Program

```
class first
    {
    public static void main(String args[])
         {
                   /*write the code here*/
         }
    }
```

Example-1: Writing First Java Program

```
class first
   {
   public static void main(String args[])
       {
       System.out.println(“This is my first java program");
       }
   }
```

Example-1: Writing First Java Program

```
class first
    {
    public static void main(String args[])
         {
         System.out.println(“This is my first java program");
         }
    }
```

Save as: first.java

Compile: javac first.java

Run: java first

Output: This is my first java program

Why main() is Public in Java?

- public keyword is an access modifier.
- Access specifier allows the programmer to control the visibility of class members.
- public members can be accessed by code outside the class in which it is declared.
- That means public members are visible to all
- The function main() must be declared as public, since it must be called by code outside of its class when the program is started.

Why main() is static in java?

- static is a keyword

- Any method declared as static, is known as static method.

- The keyword static allows main() to be called without having to instantiate a particular instance of the class.

- This is a necessary requirement since main( ) is called by the JVM before any objects are made

Arguments of main()

- In main(), there is only one parameter String args[].
- String is a class and args is array of instances of the class String.
- Here, args receives any command-line arguments present when the program is executed

Example-2

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

Save as: hello.java

Compile: javac hello.java

Run: java hello

Output: Hello World

Example-3

```
class sum{
public static void main(String args [ ])
   {
         int n1, n2, s;
         n1 = 10;
         n2=20;
         s=n1+n2;
         System.out.println("The sum of two numbers is:” +s);
   }
}
```

Output: The sum of two numbers is: 30

Simple Printing Syntax

C	JAVA
printf (“Hello World”);	System.out.println(“Hello World”);
printf(“%d”,n);	System.out.println(n);
printf(“the number is %d”,n);	System.out.println(“the number is”+ n);

Simple Printing Syntax

For getting the Output:

The sum of 5 and 4 is: 9

In C:

```
printf(“ The sum of  %d and %d  is:  %d ”, n1, n2, sum);
```

In Java:

```
System.out.println(……  ???
```

Example Printing Syntax

C	JAVA
printf(“The sum of %d and %d is: %d ”,n1,n2, sum);	System.out.println(“The sum of ” +n1 + “and” + n2 + “is: ” +sum);

Homework Questions

1. Write a Java program to check whether a number is even or odd.

2. Write a Java program to find the sum of digits of a number.

3. Write a Java program to print first n natural numbers

Reading Inputs

Reading Integer Values

C	JAVA

#include<stdio.h>
scanf(“%d”,&n);
	import java.util.Scanner;
	Scanner sc = new Scanner(System.in);
	int n = sc.nextInt();

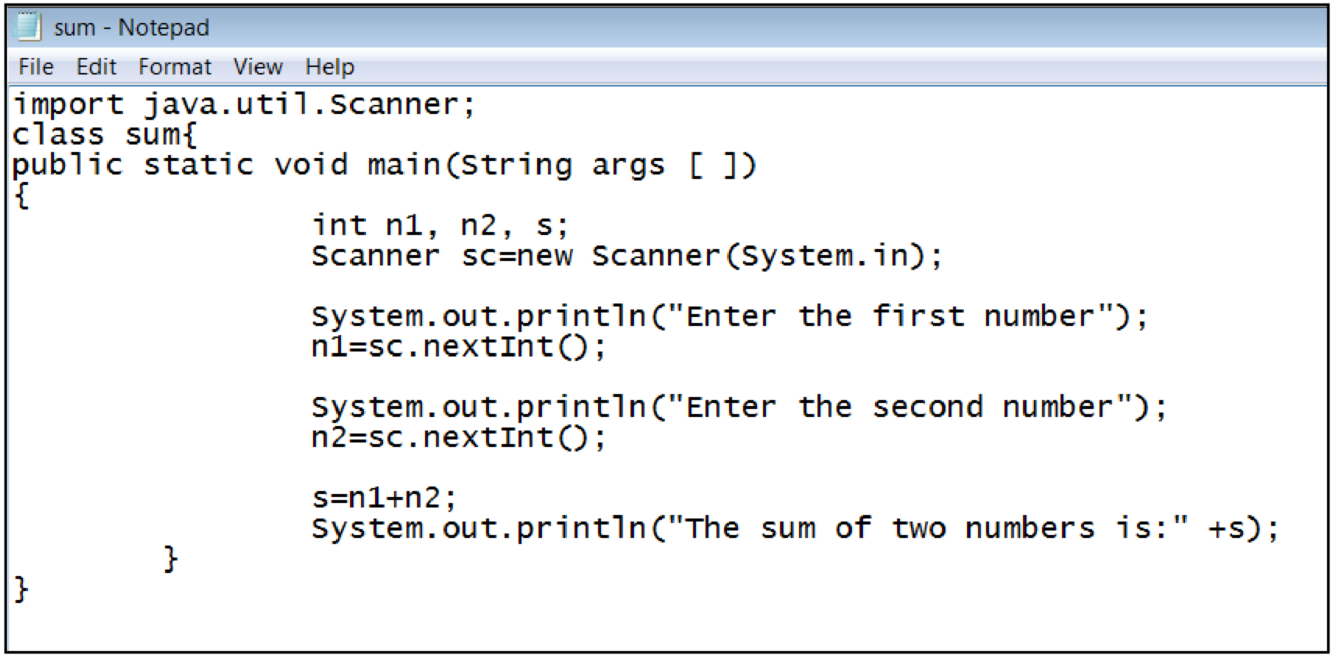
Example-1

```
import java.util.Scanner;

class readnum {
            public static void main(String args[])
            {
            int n;
            Scanner sc = new Scanner(System.in);

            System.out.println("enter a number");
            n = sc.nextInt();

            System.out.println(“the entered number is”+n);
            }
      }
```

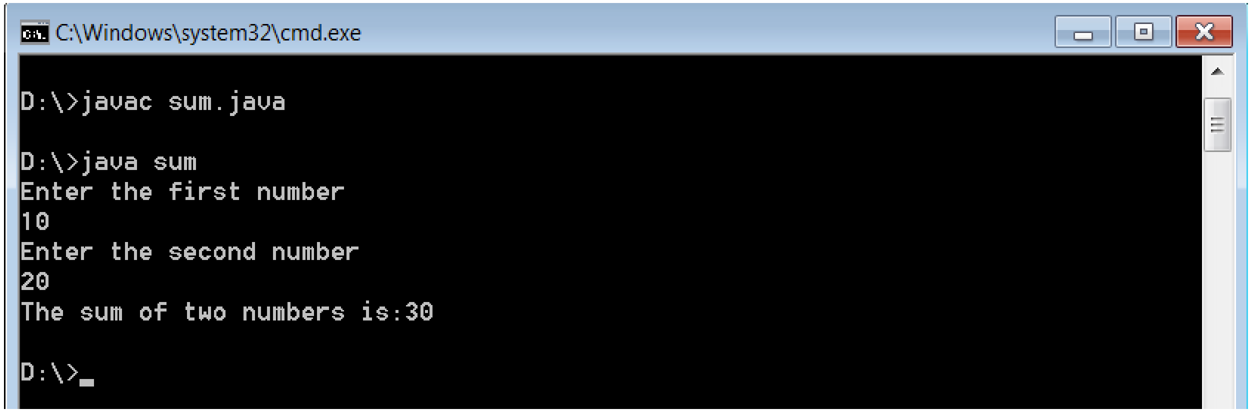
Example-2: Reading Multiple Inputs

```
import java.util.Scanner;

class readnum {
        public static void main(String args[])
                {
                 int n1, n2;
                Scanner sc = new Scanner(System.in);

                System.out.println("enter the first number");
                n1 = sc.nextInt();

                System.out.println("enter the 2nd number");
                n2 = sc.nextInt();
                        /*program logic*/
                        ……..
                }
        }
```

Example



```
import java.util.Scanner;
class sum{
public static void main(String args [ ])
{
                int n1, n2, s;
                Scanner sc=new Scanner(System.in);

                System.out.println("Enter the first number");
                n1=sc.nextInt();

                System.out.println("Enter the second number");
                n2=sc.nextInt();

                s=n1+n2;
                System.out.println("The sum of two numbers is:" +s);
        }
}
```



```

D:\>javac sum.java

D:\>java sum
Enter the first number
10
Enter the second number
20
The sum of two numbers is:30

D:\>_
```

Reading float and double values

float values	float f = sc.next**F**loat();
double values	double d= sc.next**D**ouble();

Reading String and character

string type values	String str = sc.nextLine();
character	char ch = sc.**next().charAt(0);**

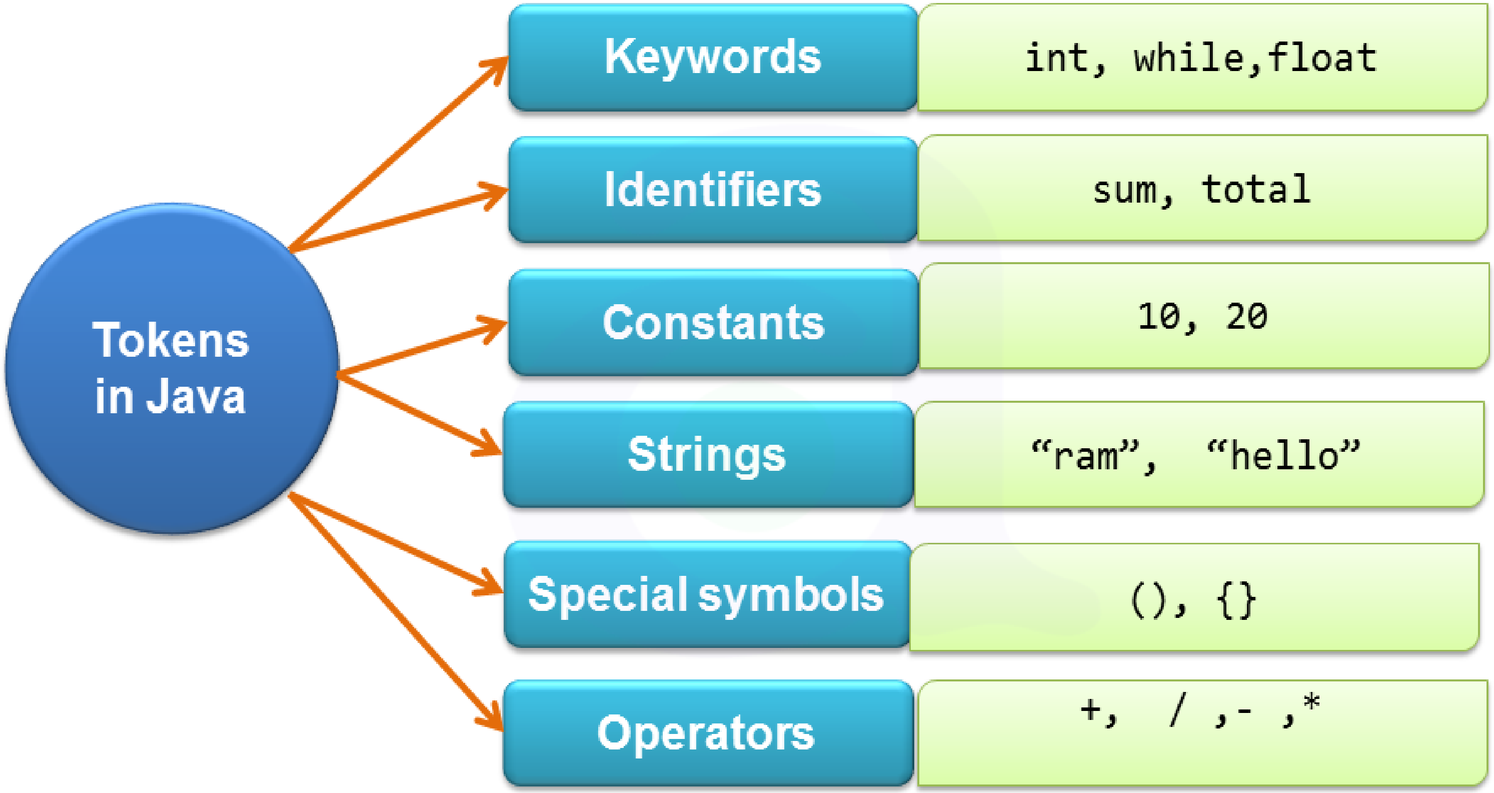
- To read a character in java, the **next()** method is used followed by **charAt(0)**.

- The next() method returns the next token/ word in the input as a string
- chatAt() method returns the first character in that string.

Example-3: reading a String

```
import java.util.Scanner;

class readtext {
                public static void main(String args[])
                {
                 String str;
                Scanner sc = new Scanner(System.in);

                System.out.println("enter a string");
                str= sc.nextLine();

                System.out.println(“the entered string is ”+str);
                }
        }
```

Tokens



Tokens in Java

Keywords	int, while,float
Identifiers	sum, total
Constants	10, 20
Strings	“ram”, “hello”
Special symbols	(), {}
Operators	+, / ,- ,*

Keywords

abstract
assert
boolean
break
byte
case
catch
char
class
const
continue
default
do
double
else
enum
extends
final
finally
float
for
goto
if
implements
import
instanceof
int
interface
long
native
new
package
private
protected
public
return
short
static
strictfp
super
switch
synchronized
this
throw
throws
transient
try
void
volatile
while

Identifiers

- Identifiers are the names of variables, methods, classes, packages and interfaces

- Note: Always choose the identifier a meaningful word.

Rules for creating identifiers:

- Should be single word.
- Should start with a letter (alphabet) or underscore or $ symbol.
- Should not be a keyword of Java.
- Should not start with a digit but digit can be in the middle or at the end.
- Identifiers are **case-sensitive.**
- Can contain all uppercase letters or lowercase letters or a mixture.

Variables

local variable:

- A variable declared inside the method is called local variable.

instance variable:

- A variable declared inside the class but outside the method, is called instance variable . It is not declared as static.

static variable:

- A variable which is declared as static is called static variable. It cannot be local.

/*   Details will be discussed in next module */

Example

```
class  A
     {
     …….
     int data=50;        //instance variable

     static int m=100;      //static variable


     …….
     void method()
               {
               int n=90;      //local variable
               }
     }  //end of class
```

Data Types

- **Primitive data types**
- **Non-primitive data types**

Primitive Data Types

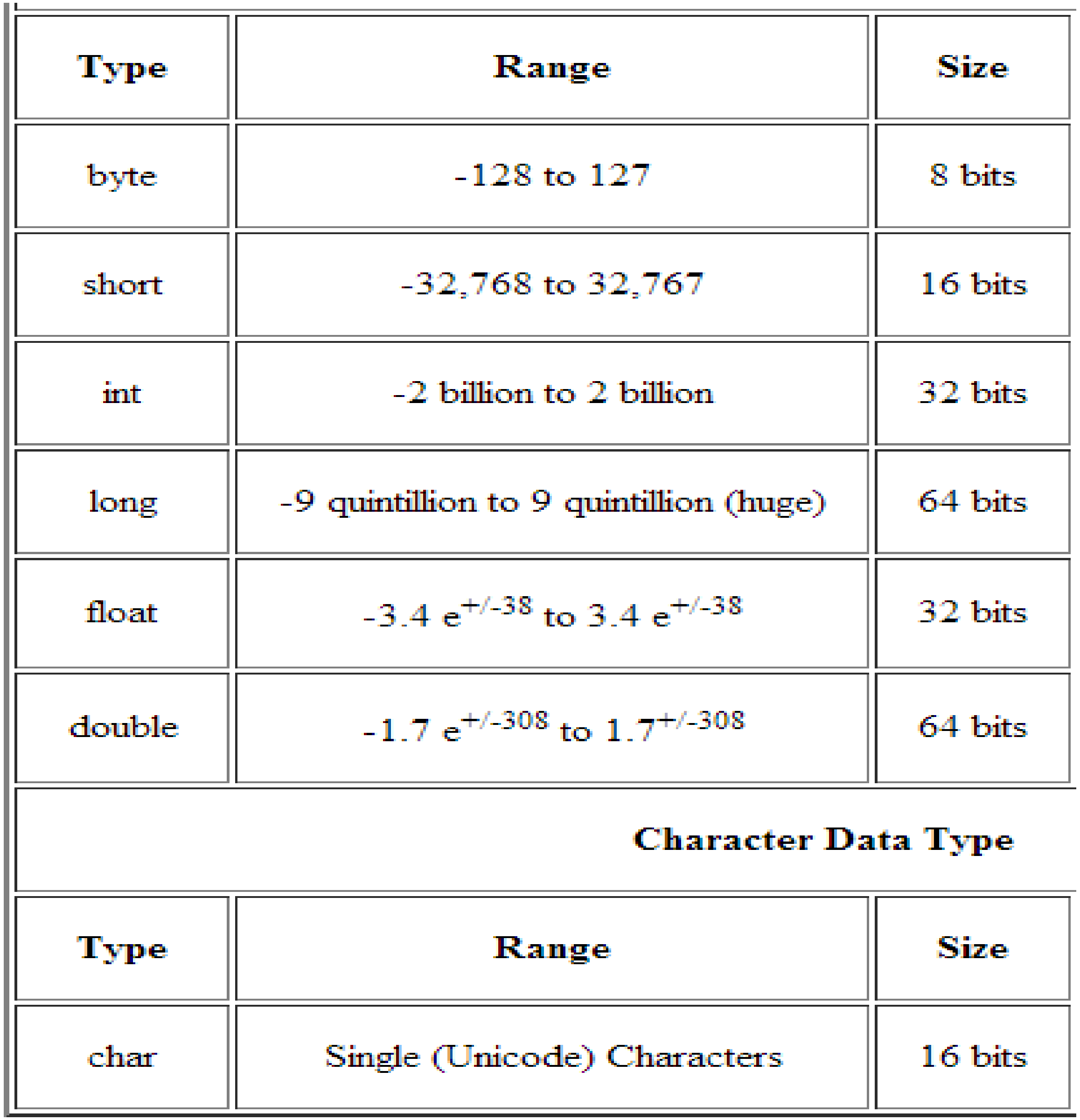
- **Integers: byte**, **short**, **int**, **long**, which are whole valued signed (no unsigned in Java) numbers.

- **Floating-point numbers: float** and **double**

- **Characters: char**

- **Boolean: boolean**
  - a special type for representing true/false values.

Primitive Data Types:



Type	Range	Size
byte	-128 to 127	8 bits
short	-32,768 to 32,767	16 bits
int	-2 billion to 2 billion	32 bits
long	-9 quintillion to 9 quintillion (huge)	64 bits
float	-3.4 $e^{+/-38}$ to 3.4 $e^{+/-38}$	32 bits
double	-1.7 $e^{+/-308}$ to $1.7^{+/-308}$	64 bits
Character Data Type		
Type	**Range**	**Size**
char	Single (Unicode) Characters	16 bits

Cont…

Why char in java takes 2 bytes ?

- As compared to the 8 bit ASCII scheme in the range of 0 to 127, the **Unicode** uses 16 bits in the range 0 to 65,535.
- i.e. in Unicode, character holds 2 byte, so java also uses 2 byte for characters.
- Since Java is designed to allow programs to be written for worldwide use, it uses Unicode to represent characters.

Cont…

Non Primitive data types: (Reference/Object Data Types):

- Class objects and various type of array variables, etc

Operators

- Arithmetic Operators
- Relational Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Miscellaneous Operators

/*same concepts as C Programming */

// Self study from book

Conditional and Loop Statements

- If-else statements
- Switch-case

- while
- do-while
- for

/*same concepts as C Programming */

/*   revise form C programming */

Cont…

Example:

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

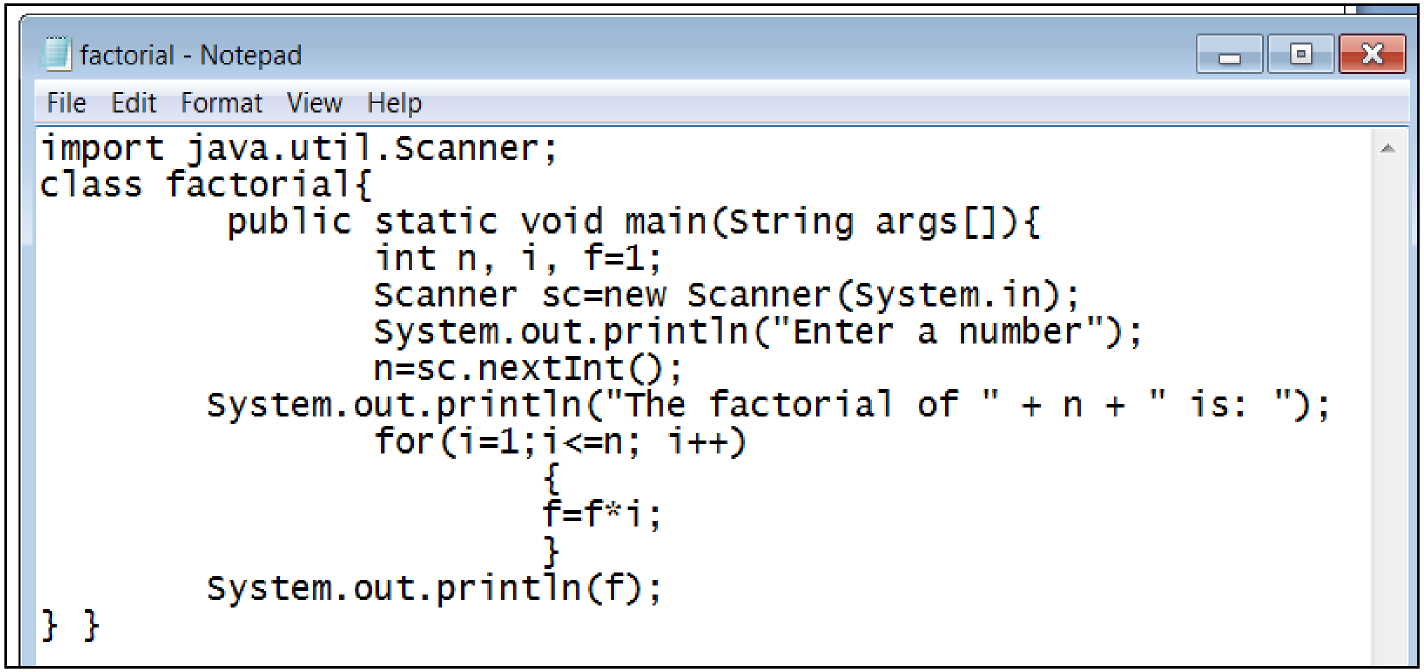
Other format :

```
for(int i=1;i<=10;i++)     //not allowed in C programming but valid in java
{
        System.out.println(i);
}
```

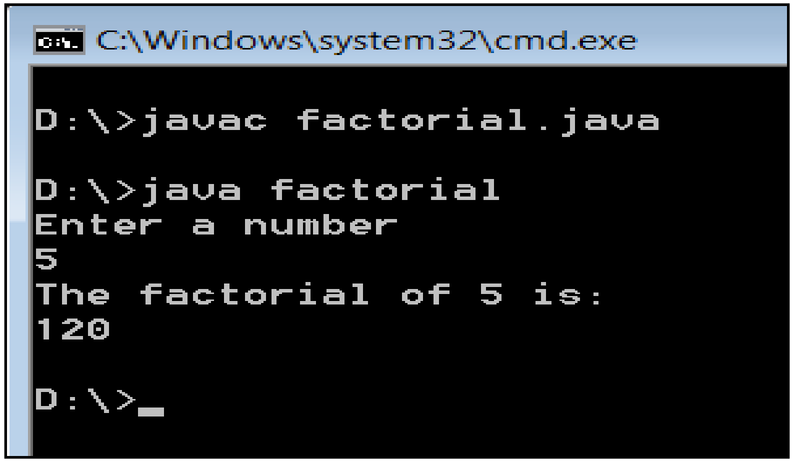
Example-1:

```
// factorial of any user entered number
import java.util.Scanner;
class factorial{
   public static void main(String args[]){
        int n, i, f=1;
        Scanner sc=new Scanner(System.in);
        System.out.println(“Enter a number”);
        n=sc.nextInt();
  System.out.println("The factorial of" + n + “ is: " );
        for(i=1;i<=n; i++) {
                f=f*i;
                }
  System.out.println(f);
} }
```

Cont…



```
import java.util.Scanner;
class factorial{
         public static void main(String args[]){
                int n, i, f=1;
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter a number");
                n=sc.nextInt();
        System.out.println("The factorial of " + n + " is: ");
                for(i=1;i<=n; i++)
                        {
                        f=f*i;
                        }
        System.out.println(f);
} }
```



```
D:\>javac factorial.java

D:\>java factorial
Enter a number
5
The factorial of 5 is:
120

D:\>_
```

Example-2: for loop and if statement

```
//factors of a number
class factor
{
    public static void main(String args[]){
            /*enter a number…….*/

   System.out.println("The factors of " + n + " are: ");
            for(int i=1;i<=n; i++)
            {
                    if(n%i==0)
                    System.out.println(i);
            }
```

Example-3: switch-case statements

```
// print the respective day on any user entered numbers
class day {
    public static void main(String args[]) {
            /*enter a number…….*/
    switch(n1){
            case 1:
                    System.out.println("Monday");
                    break;
                    ……
            case 7:
                    System.out.println("Sunday");
                    break;
            default :
                    System.out.println("Invalid option");
} } }
```

Example-4

```
/*Q. Write a menu driven program to perform simple calculator operations
(+, -, *, /) on 2 user entered numbers*/

class menu {
            public static void main(String args[]) {
                      /*enter num1 and num2.........*/
           System.out.println("enter the operation: + - * /  ");
           char ch = sc.next().charAt(0);
switch(ch){
           case  '+':
                      //find sum of num1 and num2 and print
           break;
           …..
           …..
           default :
           System.out.println(“Invalid option”);
}
```

Type Casting

- When the value of one data type is assigned to another type,
  - the two types might not be compatible with each other

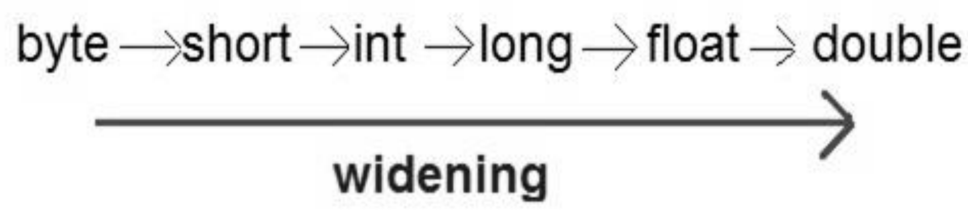
- **Widening** or Automatic (**Implicit**) Type Conversion

- **Narrowing** or **Explicit** Conversion

Implicit Type Conversion

- When two data types are automatically converted

- This happens when:
  - The two data types are compatible
  - The destination type is larger than the source type.



byte → short → int → long → float → double

widening

Example: Implicit Type Conversion

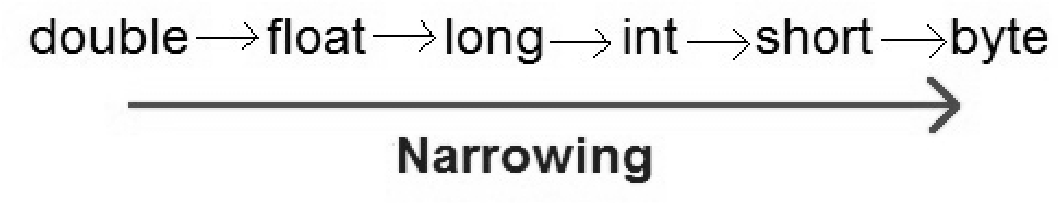
```
class test
{
    public static void main(String args [] )
            {
            int i = 100;

            long l = i; //automatic type conversion
            float f = i; //automatic type conversion
            System.out.println("Int value: "+i);
            System.out.println("Long value: "+l);
            System.out.println("Float value: "+f);
            }
}
```

Output:

```
Int value: 100
Long value: 100
Float value: 100.0
```

Explicit Type Conversion

- If a value of larger data type is required to assigned to a smaller data type explicit type casting or narrowing is performed.



double → float → long → int → short → byte

Narrowing

Example-1:

```
class Test
{
        public static void main(String args[])
        {
                char ch = 'c';
                int num = 97;
                ch = num;        //Correct code: ch = (char)num;
                System.out.println(ch);
        }
}
```

Output:

```
7: error: incompatible types: possible lossy conversion from int to char
ch = num;
     ^
1 error
```

Example-1:

```
class Test
{
        public static void main(String args[])
        {
                char ch = 'c';
                int num = 97;
                ch = (char)num;
                System.out.println(ch);
        }
}
```

Output:

```
a
```

Example-2:

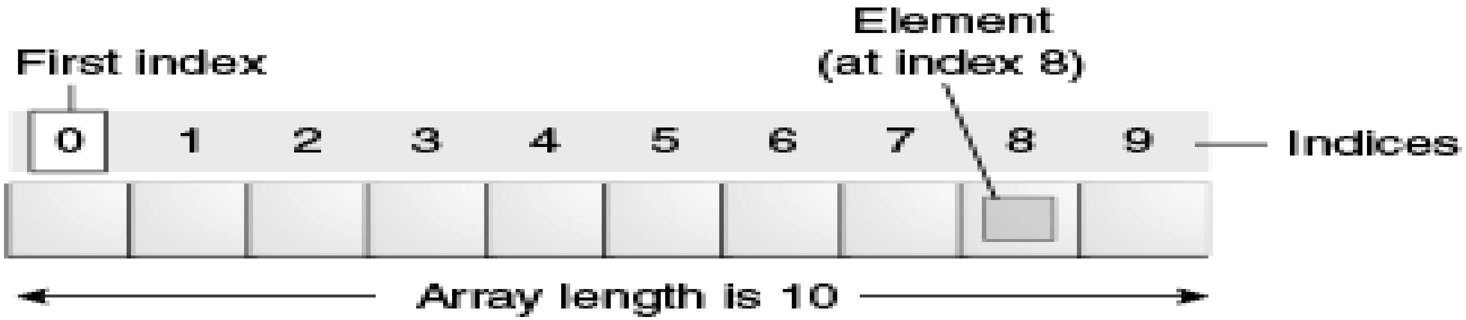
```
class Test
{
        public static void main(String args[])
        {
                double d = 100.04;
                int i = (int)d;
                System.out.println("Double value: " +d);
                System.out.println("Int value: "+i);  //fractional part lost
        }
}
```

Output:

```
  Double value: 100.04
  Int value: 100
```

Arrays

One-Dimensional Arrays



Syntax:

C	Java

int a[];
//invalid in C
	int a[];
	// correct but only array variable is created
	
	a=new int[3] //allocates the memory

int a[3]; //correct format in C	int a[]=new int[5];

Cont…

Example:

```
int a[]= new int[3];

          or

int a[]={1,2,3};
```

Example

```
class Testarray
{
   public static void main(String args[])
            {
            int a[]=new int[5];//declaration and instantiation
            a[0]=10;//initialization
            a[1]=20;
            a[2]=30;
            a[3]=40;
            a[4]=50;
```

Cont…

```
//printing the array contents

for(int i=0; i<a.length; i++)       // .length finds the size of the array a
            System.out.println(a[i]);
   }
}
```

```
Output:
10
20
30
40
50
```

Example-1: Array Initialization

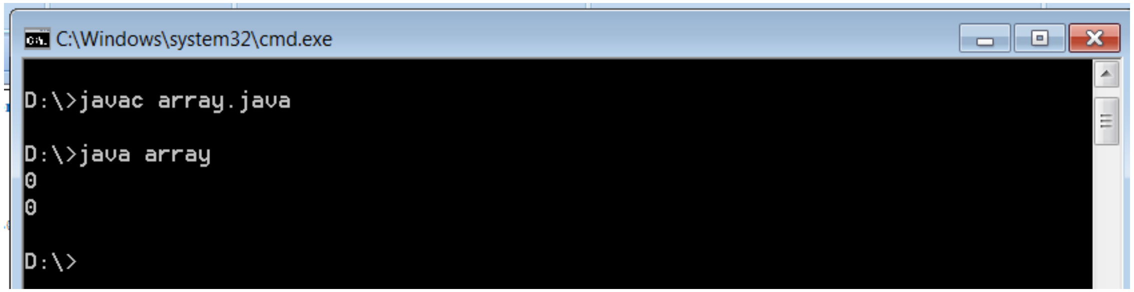
```
class array
{
    public static void main(String args[])
              {
              int mark[]={55,65};
              System.out.println(mark[0]);
              System.out.println(mark[1]);
              }
    }
```

Example-2: Another Approach

```
class array
{
    public static void main(String args[])
            {
            int mark[]=new int[2];
            mark[0]=75;
            mark[1]=86;
            System.out.println(mark[0]);
            System.out.println(mark[1]);
            }
}
```

Example-3: Uninitialized Array

```
class array
{
  public static void main(String args[])
         {
         int mark[]=new int[2];

         System.out.println(mark[0]);
         System.out.println(mark[1]);
         }
}
```



```
D:\>javac array.java

D:\>java array
0
0

D:\>
```

Runtime Entry of Elements into Array

In C:

```
main(){
        int a[5];
        int i;
        printf(“Enter the array elements\n”);
        for(i=0;i<5;i++)
                  {
                  scanf(“%d”, &a[i]);
                  }
        printf(“The array elements are\n”);
        for(i=0;i<5;i++)
                  {
                  printf(“%d\t”,a[i]);
                  }
        }
```

Example-4: Runtime Entry of Elements into Array

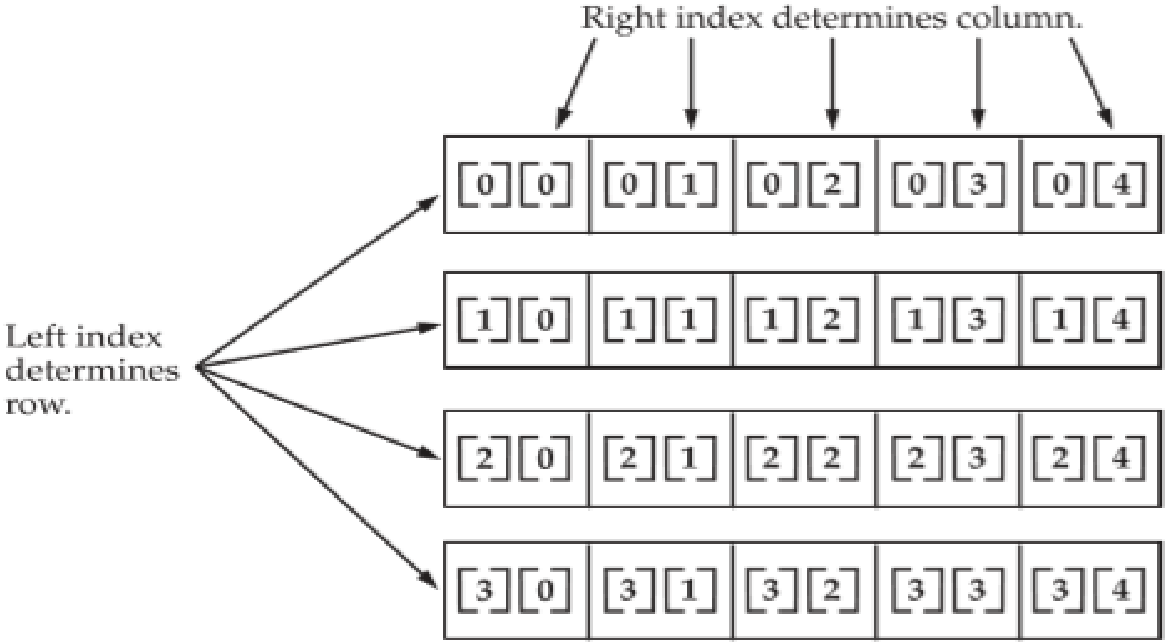
```
import java.util.Scanner;
class array {
  public static void main(String args[]) {
        int i;
        Scanner sc = new Scanner(System.in);
        System.out.println("enter the size of the array ");
        int n= sc.nextInt();
        int arr[]=new int[n];
        System.out.println("Enter the array elements");
        for(i=0;i<n;i++) {
                arr[i]=sc.nextInt();
                }
```

Cont…

```
//Displaying elements of the array

System.out.println("the array elements are");
    for(i=0;i<n;i++)
    {
    System.out.println(arr[i]);
    }
}
}
```

Multidimensional Arrays

Multi-dimensional Arrays



Right index determines column.

Left index determines row.

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]

Given: int twoD [ ] [ ] = new int [4] [5] ;

Cont…

Syntax:

C	Java

int m1[][];
//invalid in C
	int m1[][];
	// valid but only variable is created
	
	m1=new int[3][3] //allocates the memory

int m1[3][3];	int m1[][]=new int[3][3];

Cont…

Example:

```
int m1[][]= new int[3][3];
```

or

```
int m1[][]={{1,2,3},{4,5,6},{7,8,9}};
```

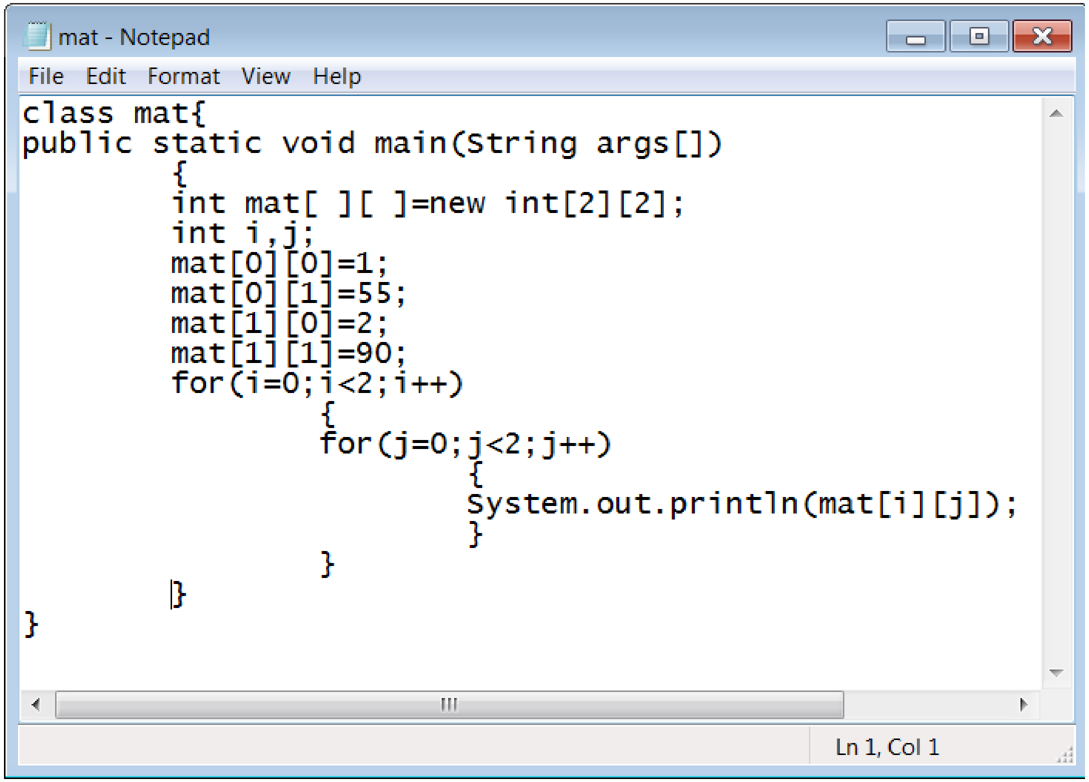
Example-1: Matrix Initialization

```
class mat
{
  public static void main(String args[])
       {
       int mat[ ][ ]={{1,65},{2,78}};

       System.out.println(mat[0][0]);
       System.out.println(mat[0][1]);
       System.out.println(mat[1][0]);
       System.out.println(mat[1][1]);
}
}
```

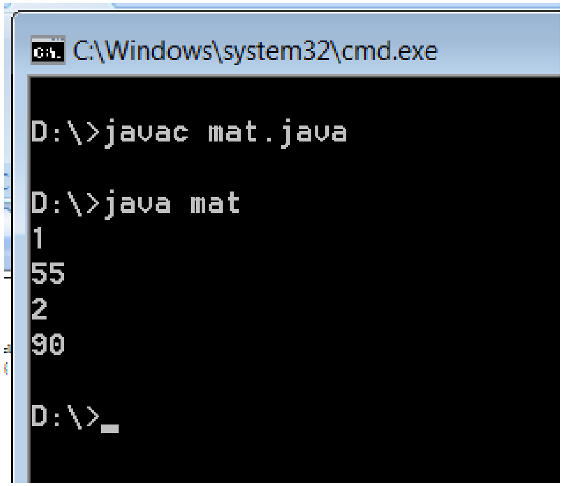
Example-2: Printing Matrix Elements

```
int mat[ ][ ]=new int[2][2];
mat[0][0]=1;
mat[0][1]=55;
mat[1][0]=2;
mat[1][1]=90;
for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
        System.out.println(mat[i][j]);
        }
    }
```

Example-2: Cont…



```
class mat{
public static void main(String args[])
        {
        int mat[ ][ ]=new int[2][2];
        int i,j;
        mat[0][0]=1;
        mat[0][1]=55;
        mat[1][0]=2;
        mat[1][1]=90;
        for(i=0;i<2;i++)
                {
                for(j=0;j<2;j++)
                        {
                        System.out.println(mat[i][j]);
                        }
                }
        }
}
```



```
D:\>javac mat.java

D:\>java mat
1
55
2
90

D:\>_
```

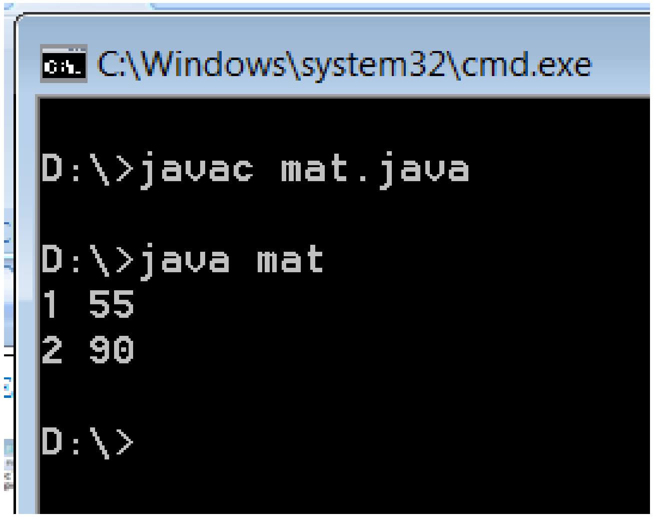
What to do for printing :

```
1    55
2     90
```

? ? ?

Cont…

```
for(i=0;i<2;i++)
    {
     for(j=0;j<2;j++)
             {
             System.out.print(mat[i][j]+ " ");
             }
    System.out.print("\n");
    }
} //end of main
} //end of class
```



```
D:\>javac mat.java

D:\>java mat
1 55
2 90

D:\>
```

Example-3: Runtime Entry of Elements into Matrix

```
import java.util.Scanner;
class matrix1
{
        public static void main(String arg[])
                {
                int r, c, i, j;
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter the size of the matrix");
                r=sc.nextInt();
                c=sc.nextInt();
                int a[][]=new int[r][c];
```

Cont…

```
System.out.println("Enter the elements of the matrix");

for(i=0;i<r;i++)
   {
   `for(j=0;j<c;j++)
          {
          a[i][j]=sc.nextInt();
          }
   }
```

Cont…

```
for(i=0;i<r;i++)
    {
     for(j=0;j<c;j++)
              {
              System.out.print(a[i][j]+ " ");
              }
    System.out.print("\n");
    }
} //end of main
} //end of class
```

Homework Questions

1. Write a Java program to test whether an entered number is prime or not.
2. Write a Java program to display the numbers between a range divisible by 3 and 5.
3. Find the largest element in an array.
4. Search whether an element is present in an array.
5. Sort the elements of the array in ascending order.
6. Find sum of all matrix elements.
7. Print all diagonal (left diagonal only) elements of a matrix.

Note: The elements of array and matrix should be entered at runtime for all programs.