



JAVA PROGRAMMING

Chap 1 : Java Fundamentals

Program: Master of Computer Application (MCA)				Semester: I	
Course: Java Programming				Code:	
Teaching Scheme				Evaluation Scheme	
Lecture (Hours per week)	Practical (Hours per week)	Tutorial (Hours per week)	Credit	Internal Continuous Assessment (ICA) (Marks - 50)	Term End Examinations (TEE) (Marks - 100)
2	4	0	4	Marks Scaled to 50	Marks Scaled to 50
Prerequisite: NA					
Course Objective This course will impart knowledge of object-oriented programming, building graphical user interface and database connectivity using Java.					
Course Outcomes After completion of the course, students will be able to - <ol style="list-style-type: none"> 1. Implement programs using object oriented programming paradigm 2. Implement programs using collection and generics concepts 3. Develop GUI application with database connectivity 					

Detailed Syllabus:		
Unit	Description	Duration
1	Java Fundamentals Overview of Java, Using Blocks of code, Lexical Issues, Java Class Libraries, Data Types, Variables and Arrays, Operators, Control Statements, Command Line Arguments.	02
2	Classes and Methods Class fundamentals, Declaring Objects, Constructors, Methods, Overloading of methods, Access control, Static and final variables.	04
3	Inheritance Inheritance Basics, method overriding, using abstract classes, using final with inheritance.	04
4	Packages and Interfaces Packages, Access Protection, importing packages, , Interfaces: Defining an Interface, Implementing Interfaces , Applying Interfaces, Variables in Interfaces.	03
5	Exception Handling Exception handling fundamentals, exception types, uncaught exceptions, using try and catch, throw, throws, finally, Java's built-in exceptions, creating your own exceptions.	02
6	Programs using String Handling String Constructors, Special String operators, Character Extraction, String Comparison, Searching Strings and Modifying Strings, Buffer class and its methods.	02
7	Generics and Collections	05

	Generics: Introduction, A Generic class with Type Parameters, General Form of a Generic class, Bounded Types, Using wildcard arguments, Creating a Generic Method, Generic class Hierarchies, Collection: Collection Framework, ArrayList class, List Iterator interface, Linked List class, TreeSet class	
8	GUI design and Event Handling using Java FX Introduction, JavaFX Architecture, application structure, JavaFX, Text, Effect, Anim, UI controls. Types of Events, Processing Events in JavaFX, Event Delivery Process, Event Handlers.	05
9	Java and Database Programming JDBC Architecture, Types of Drivers, JDBC components, JDBC classes and Interfaces, steps for querying the database with JDBC, Database connection, querying and updating database tables, passing parameters to a statement.	03
	Total	30
Text Books:		
1. Herbert Schildt, <i>Java The Complete Reference</i> , 11 th Edition, Oracle Press, 2020.		
2. Sergey Grinev, <i>Mastering JavaFX10</i> , Packt Publishing, 2018.		
Reference Books:		
1. Cay Horstmann, <i>Core Java Volume I- Fundamentals</i> , Pearson Education Inc., 2020.		
2. Carl Dea, Gerrit Grunwald, José Pereda, Sean Phillips, Mark Heckler, <i>JavaFX 9 by Example</i> , 3 rd Edition, Apress, 2017.		

- Java is a programming language and a platform.
- Java is a high level, robust, object-oriented and secure programming language.
- Java was developed by Sun Microsystems (which is now the subsidiary of Oracle) in the year 1995.
- James Gosling is known as the father of Java.
- Before Java, its name was Oak. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

- There are mainly 4 types of applications that can be created using Java programming:

1) Standalone Application : Also known as desktop applications or window-based applications are traditional software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.

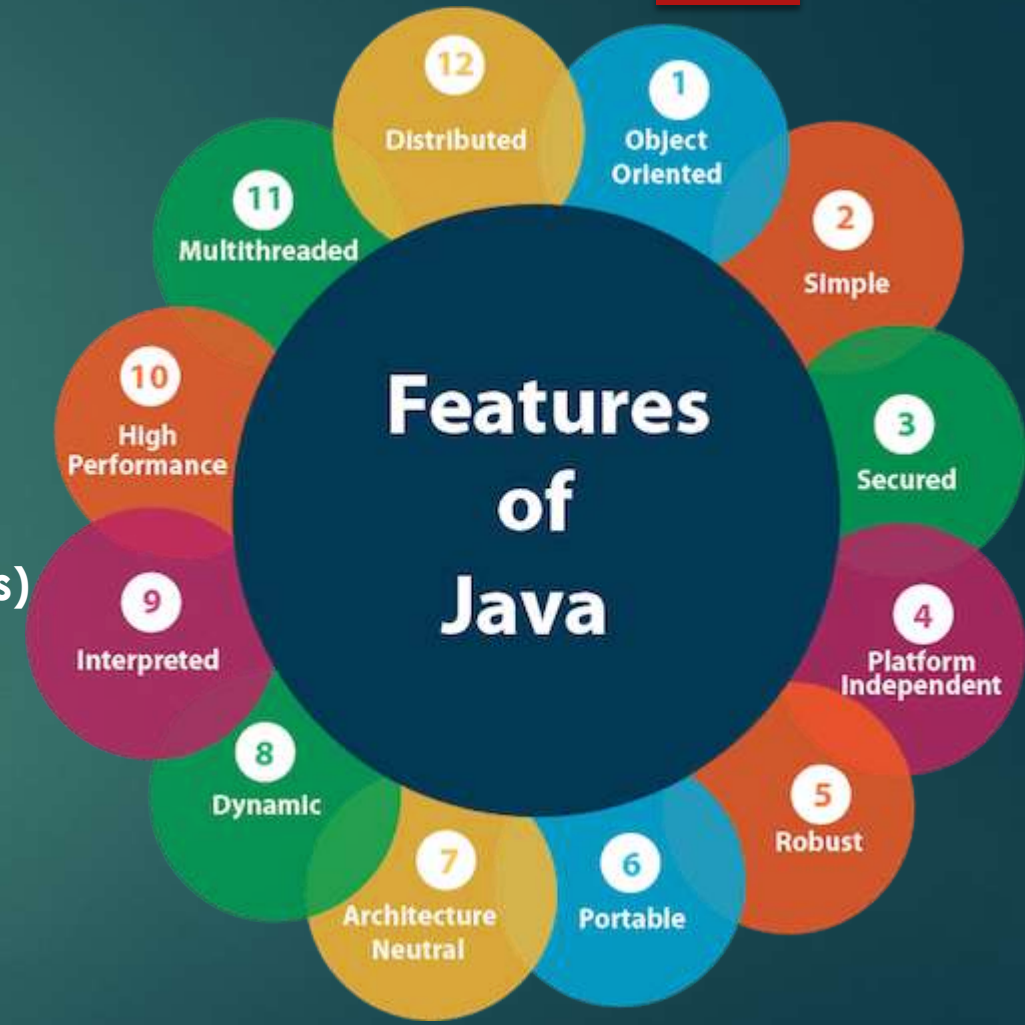
2) Web Application : An application that runs on the server side and creates a dynamic page is called a web application. Currently, Servlet, JSP, Struts, Spring, Hibernate, JSF, etc. technologies are used for creating web applications in Java.

3) Enterprise Application : An application that is distributed in nature, such as banking applications, etc. is called an enterprise application. It has advantages like high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.

4) Mobile Application : An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

Features of Java

- 1) Simple - (No pointers, no operator overloading, no garbage collection, no multiple inheritance)
- 2) Object-Oriented
- 3) Portable - (bytecode)
- 4) Platform independent - (bytecode)
- 5) Secured - (no explicit pointer)
- 6) Robust - (strong MM Mgmt, Exception Handling, Auto Garbage Collection, no pointers)
- 7) Architecture neutral - (bytecode, fixed size of primitive datatypes)
- 8) Interpreted
- 9) High Performance
- 10) Multithreaded
- 11) Distributed - (helps to create distributed applns using RMI & EJB)
- 12) Dynamic - (uses JIT)
- 13) Garbage Collected



Basic concepts of OOPs are:

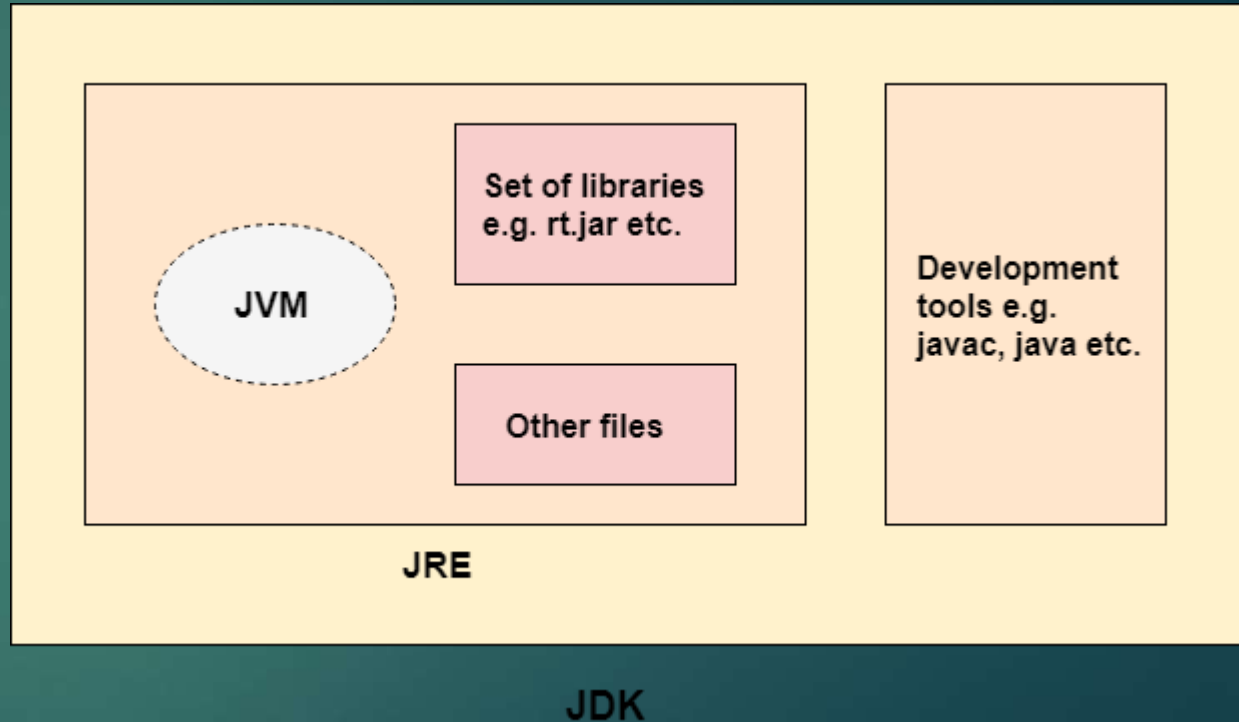
- 1) Object
- 2) Class
- 3) Inheritance
- 4) Polymorphism
- 5) Abstraction
- 6) Encapsulation

C++	Java
It is object oriented but not purely. That is a program can be written in C++ without using classes and objects.	It is purely object oriented that is no program can be written without classes and objects.
C++ is platform dependent	Java is platform independent
C++ supports the goto statement.	Java doesn't support the goto statement.
C++ supports pointers. This makes program system dependent.	Java doesn't support pointers to avoid platform dependency.
C++ supports multiple and hybrid inheritance	Java does not support multiple and hybrid inheritance. A slight implementation of the same can be done using a special feature called as Interface.
Supports operator overloading	Does not allow operator overloading
Supports 3 access specifiers : public, protected and private	Supports 5 access specifiers : public, protected, private, default and private protected
Destructors are used in C++	Java is garbage collected that is the objects are automatically destroyed once their use is over
C++ doesn't have exception handling	Java has exception handling
Doesn't support multithreading	Supports multithreading

[illegible]

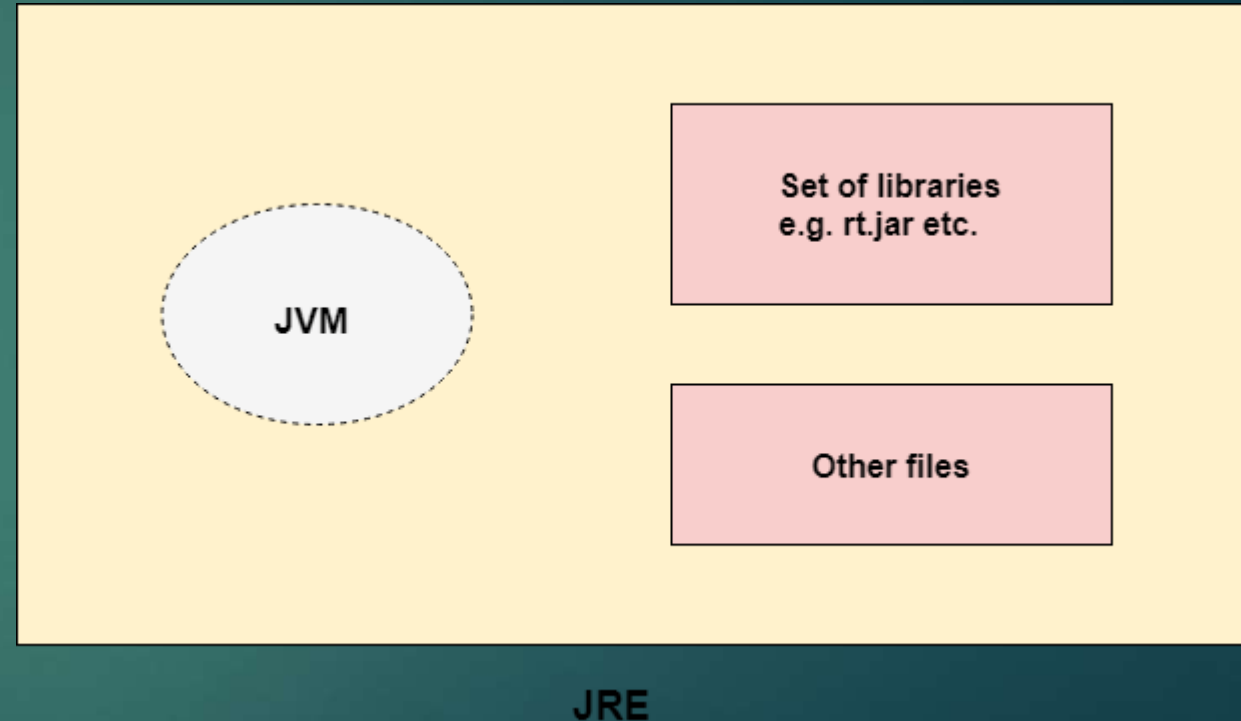
Java Development Kit (JDK)

- ▶ JDK is an acronym for Java Development Kit.
- ▶ The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets.
- ▶ It physically exists. It contains JRE + development tools.
- ▶ The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.



Java Runtime Environment (JRE)

- ▶ JRE is an acronym for Java Runtime Environment.
- ▶ It is also written as Java RTE.
- ▶ The Java Runtime Environment is a set of software tools which are used for developing Java applications.
- ▶ It is used to provide the runtime environment.
- ▶ It is the implementation of JVM.
- ▶ It physically exists.
- ▶ It contains a set of libraries + other files that JVM uses at runtime.
- ▶ The implementation of JVM is also actively released by other companies besides Sun Micro Systems.



Java Virtual Machine (JVM)

- ▶ JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.
- ▶ JVM is an interpreter. It is the main component of Java that makes java platform independent.
- ▶ It is because the bytecode file can be taken onto any system and the JVM residing in that system will interpret and translate the byte code into machine readable code.
- ▶ The JVM performs following operation:
 - ▶ Loads code
 - ▶ Verifies code
 - ▶ Executes code
 - ▶ Provides runtime environment

Tokens of Java

- ▶ Character Set
- ▶ Keywords
- ▶ Identifiers
- ▶ Constants & variables
- ▶ Datatypes
- ▶ operators

Character Set

- ▶ Alphabets: Both lowercase (a, b, c, d, e, etc.) and uppercase (A, B, C, D, E, etc.).
- ▶ Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- ▶ Special symbols: `_`, `(`, `)`, `{`, `}`, `[`, `]`, `+`, `-`, `*`, `/`, `%`, `!`, `&`, `|`, `~`, `^`, `<`, `=`, `>`, `$`, `#`, `?`, Comma `,`, Dot `.`, Colon `:`, Semi-colon `;`, Single quote `'`, Double quote `"`, Back slash `\`.
- ▶ White space: Space, Tab, New line.

Keywords

- ▶ There are 50 keywords currently defined in the Java language.
- ▶ These keywords, combined with the syntax of the operators and separators, form the foundation of the Java language.
- ▶ **These keywords cannot be used as identifiers.**
- ▶ Thus, they cannot be used as names for a variable, class, or method.
- ▶ In addition to the keywords, Java reserves the following: **true**, **false**, and **null**. These are values defined by Java. You may not use these words for the names of variables, classes, and so on.

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

Table 2-1 Java Keywords

Identifiers

- ▶ Identifiers are used to name things, such as classes, variables, and methods.
- ▶ An identifier may be any descriptive sequence of **uppercase and lowercase letters**, **numbers**, or the **underscore** and **dollar-sign** characters. (The dollar-sign character is not intended for general use.)
- ▶ Following rules needs to be followed while creating identifiers
 1. An identifier can consist of Alphabets, digits, and two special symbols _ (underscore) and \$ (dollar)
 2. An identifier cannot start with a digit. It has to start with an alphabet or _ or \$
 3. No other special symbols apart from _ and \$ are allowed. Not even blank spaces.
 4. An identifier cannot be a keyword.
 5. It is case sensitive. That is firstName, FirstName, firstname are three different identifiers.

Identifiers

Exercise : Identify valid and invalid identifiers from the list given below –

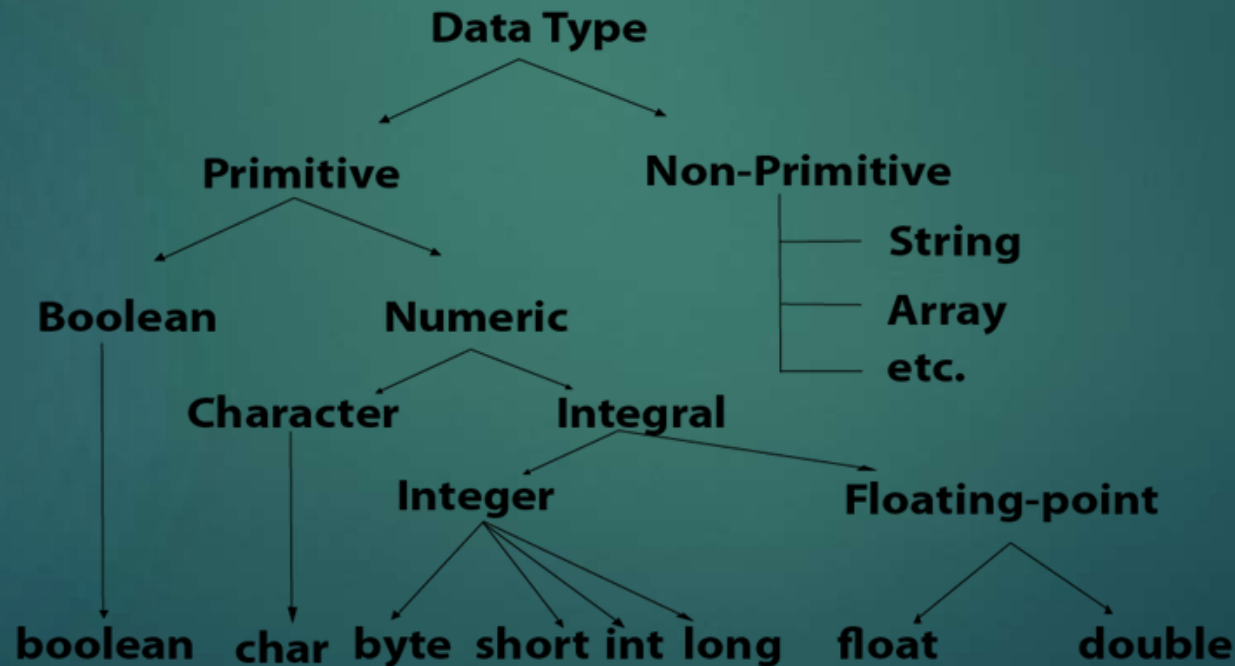
1. simple_interest
2. char
3. 3friends
4. _3\$friends
5. Simple interest
6. #3friends
7. void
8. Void
9. \$xyz

Constants and variables

- ▶ A **constant** is an entity in programming that is immutable.
 - ▶ In other words, the value that cannot be changed after assigning it.
 - ▶ In java constants can be defined by using the keyword "final" before the datatype.
 - ▶ Constants are used to declare values that remain constant like value of pi.
 - ▶ Eg : final double pi=3.14;
-
- ▶ **Variables** are containers for storing data values.
 - ▶ A variable is assigned with a data type.
 - ▶ A variable is the name of a reserved area allocated in memory.
 - ▶ In other words, it is a name of the memory location.
 - ▶ It is a combination of "vary + able" which means its value can be changed.
 - ▶ eg: **int** data=50; //Here data is a variable

Datatypes

- ▶ Data types specify the different sizes and values that can be stored in the variable.
- ▶ There are two types of data types in Java:
- ▶ **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
- ▶ **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.



Datatypes

Data Type	Default Value	Default size	Range
boolean	false	1 bit	true or false
char	'\u0000'	2 byte	'\u0000' (or 0) to '\uffff' (or 65,535 inclusive).
byte	0	1 byte	-128 to 127 (inclusive)
short	0	2 byte	-32,768 to 32,767 (inclusive)
int	0	4 byte	- 2,147,483,648 to 2,147,483,647 (inclusive)
long	0L	8 byte	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (inclusive)
float	0.0f	4 byte	-3.4e38 to 3.4e38
double	0.0d	8 byte	Its value range is unlimited

Operators

► There are many types of operators in Java which are given below:

1. Unary Operator
2. Arithmetic Operator
3. Shift Operator
4. Relational Operator
5. Bitwise Operator
6. Logical Operator
7. Ternary Operator
8. Assignment Operator

Operators

► Operator precedence

Operator Type	Category	Precedence
Unary	postfix	<i>expr</i> ++ <i>expr</i> --
	prefix	++ <i>expr</i> -- <i>expr</i> + <i>expr</i> - <i>expr</i> ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>
Relational	comparison	< > <= >= instanceof
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	
Logical	logical AND	&&
	logical OR	
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= = <<= >>= >>>=

Ternary Operator

- ▶ The operator that requires three operands is called as Ternary Operator.
- ▶ Java supports only one ternary operator `? :`
- ▶ Syntax : (condition) `?` (Value if true) `:` (value if false)

```
import java.util.*;

class Ternary{

    public static void main(String args[]){
        int a, b, large;

        Scanner sc=new Scanner(System.in);
        System.out.println("Enter 2 numbers");
        a=sc.nextInt();
        b=sc.nextInt();
        large=(a>b)?a:b;
        System.out.println("Largest number is : "+large);
    }
}
```

```
F:\Java>javac Ternary.java
```

```
F:\Java>java Ternary
```

```
Enter 2 numbers
```

```
15
```

```
34
```

```
Largest number is : 34
```


Comments

- ▶ Single line Comments - `// comments`
- ▶ Multi line comments - `/* comments */`
- ▶ Documentation comments - `/** comments */`

Input/Output in Java

Output in Java

- ▶ `System.out.println("Hello");`
- ▶ `System` – class of `Java.lang` package
- ▶ `out` – variable of class `System` corresponding to the std o/p device i.e. monitor
- ▶ `println()/print()` – static methods for displaying content

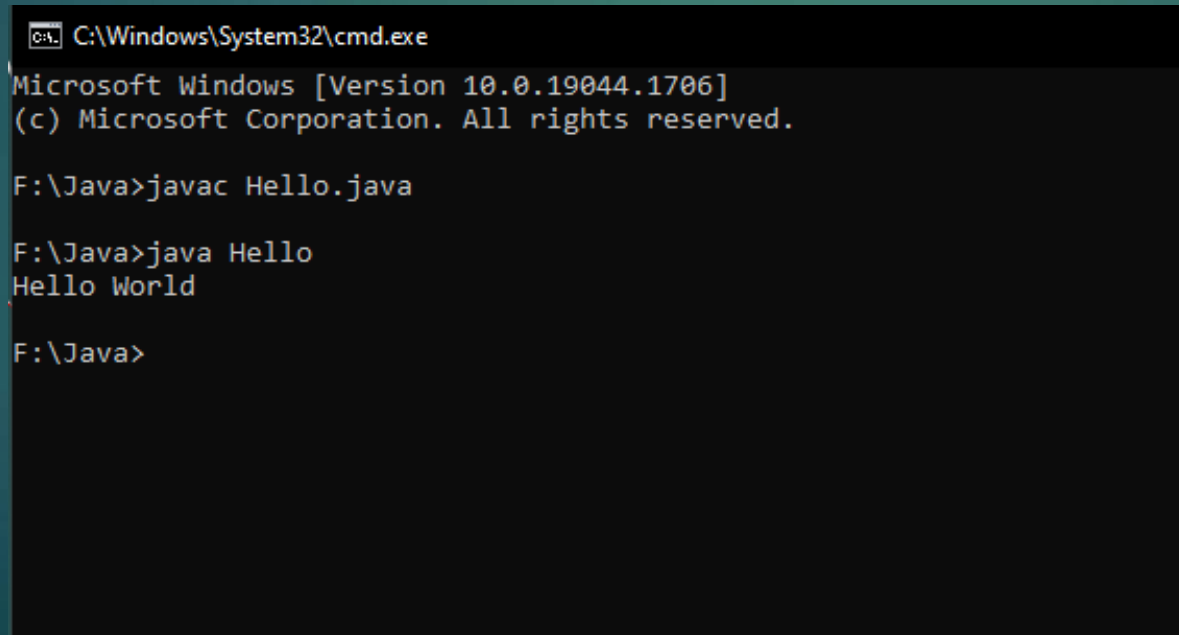
Input in Java

- ▶ `BufferedReader` – `readLine()`
- ▶ `Scanner` – `nextInt()`, `nextLong()`, `nextFloat()`, `nextDouble()`, `next()`, `nextLine()`

Simple Java Program

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

Output :



```
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.19044.1706]  
(c) Microsoft Corporation. All rights reserved.  
  
F:\Java>javac Hello.java  
  
F:\Java>java Hello  
Hello World  
  
F:\Java>
```

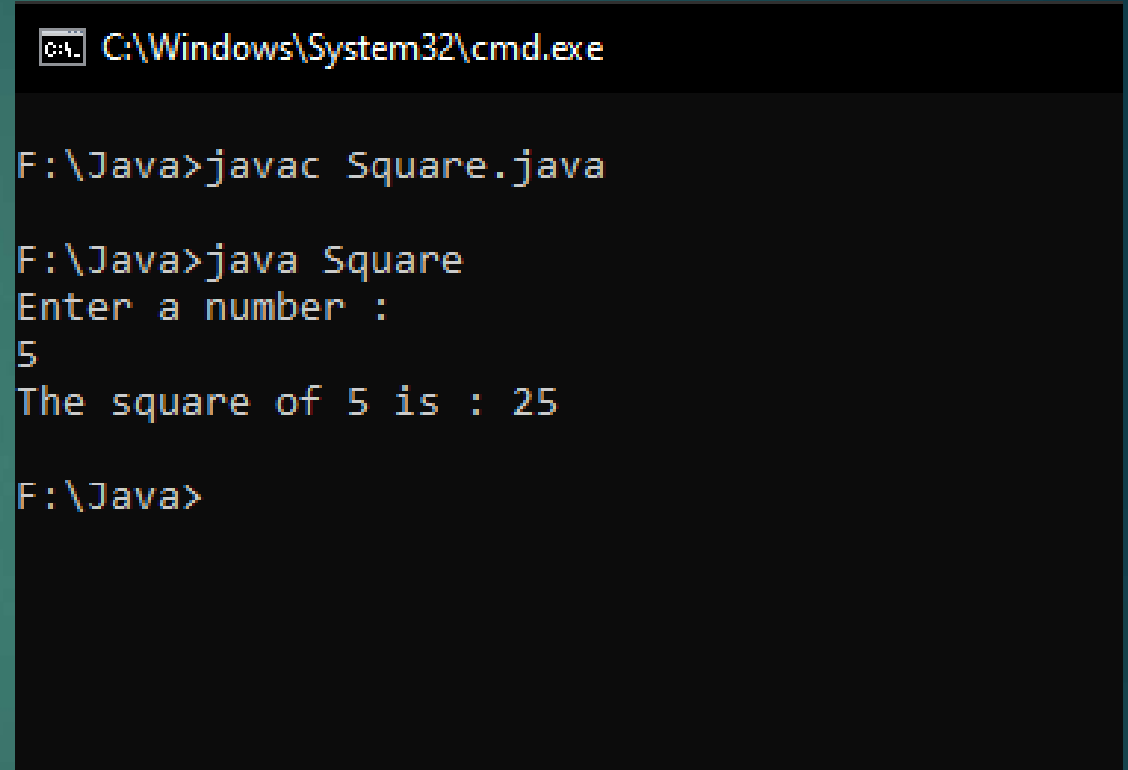
Program for accepting input (Scanner)

```
import java.util.*;

class Square{
    public static void main(String args[])
    {
        int i, res;

        Scanner sc=new Scanner (System.in);
        System.out.println("Enter a number : ");
        i=sc.nextInt();
        res=i*i;

        System.out.println("The square of " + i + " is :
" + res);
    }
}
```



```
C:\Windows\System32\cmd.exe

F:\Java>javac Square.java

F:\Java>java Square
Enter a number :
5
The square of 5 is : 25

F:\Java>
```

Program for accepting input (BufferedReader)

```
import java.io.*;

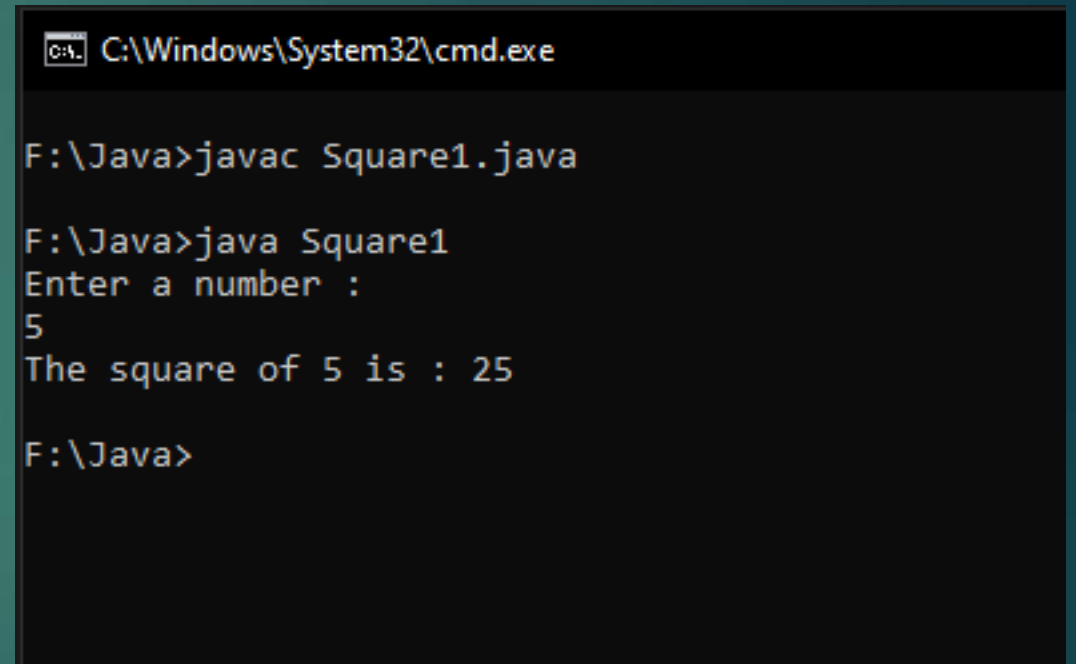
class Square{
    public static void main(String args[]) throws IOException
    {
        int i, res;

        BufferedReader br=new BufferedReader (new
InputStreamReader(System.in));

        String str;

        System.out.println("Enter a number : ");
        str=br.readLine();
        i=Integer.parseInt(str);
        res=i*i;

        System.out.println("The square of " + i + " is : " + res);
    }
}
```



```
C:\Windows\System32\cmd.exe

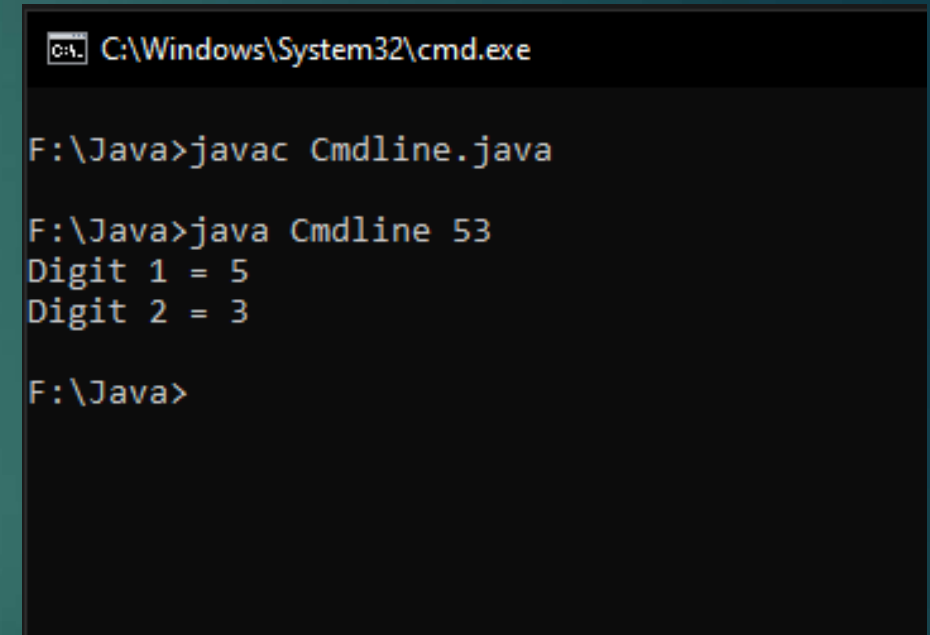
F:\Java>javac Square1.java

F:\Java>java Square1
Enter a number :
5
The square of 5 is : 25

F:\Java>
```

Program for accepting input (command line)

```
class Cmdline{  
    public static void main(String args[])  
    {  
        int n,d1,d2;  
        n=Integer.parseInt(args[0]);  
        d2=n%10;  
        d1=n/10;System.out.println("Digit 1 = "+d1+"\nDigit 2 = "+d2);  
    }  
}
```



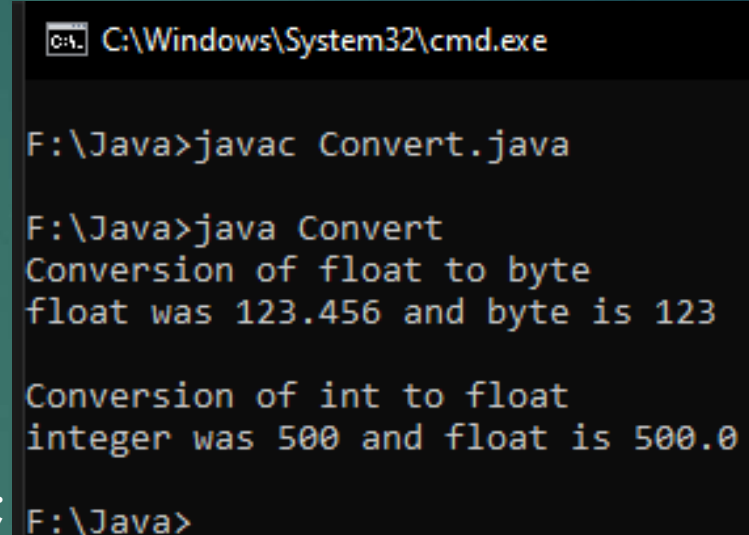
```
C:\Windows\System32\cmd.exe  
  
F:\Java>javac Cmdline.java  
  
F:\Java>java Cmdline 53  
Digit 1 = 5  
Digit 2 = 3  
  
F:\Java>
```


Typecasting in Java

- ▶ **Type casting** is a method or process that converts a data type into another data type in both ways manually and automatically.
- ▶ The automatic conversion is done by the compiler and manual conversion performed by the programmer
- ▶ Converting a lower data type into a higher one is called as **implicit conversion** or **casting down**.
- ▶ It is done automatically.
- ▶ It takes place when:
 - ▶ Both data types must be compatible with each other.
 - ▶ The target type must be larger than the source type.
- ▶ Eg : typecasting int to long
- ▶ Converting a higher data type into a lower one is called as **explicit conversion** or **casting up**.
- ▶ It is done manually by the programmer.
- ▶ If we do not perform casting then the compiler reports a compile-time error.
- ▶ Eg : typecasting float to byte
- ▶ Syntax for explicit typecasting –
(target datatype)source datatype variable
- ▶ Eg : float x;
(byte) x;

Typecasting in Java

```
class Convert{
    public static void main(String args[])
    {
        byte b;
        int i=500;
        float f=123.456f;
        System.out.println("Conversion of float to byte");
        b=(byte) f; // explicit
        typecasting
        System.out.println("float was "+f+ " and byte is "+b+"\n");
        System.out.println("Conversion of int to float");
        f=i; // implicit typecasting
        System.out.println("integer was "+i+ " and float is "+f);
    }
}
```



```
C:\Windows\System32\cmd.exe

F:\Java>javac Convert.java

F:\Java>java Convert
Conversion of float to byte
float was 123.456 and byte is 123

Conversion of int to float
integer was 500 and float is 500.0

F:\Java>
```

Control statements

- ▶ Control statements are of two types
 - ▶ Conditional statements – if-else, switch case
 - ▶ Iterative statements – for, while, do while

Conditional Statements

If-Else statements

```
import java.util.*;

class PositiveNegativeExample {
    public static void main(String[] args) {
        int n;
        Scanner sc=new Scanner (System.in);
        System.out.println("Enter n : ");
        n=sc.nextInt();
        if(n>0){
            System.out.println("POSITIVE");
        }
        else if(n<0){
            System.out.println("NEGATIVE");
        }
        else{
            System.out.println("ZERO");
        }
    }
}
```

Switch statements

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int day;
        Scanner sc=new Scanner (System.in);
        System.out.println("Enter the day number : ");
        day=sc.nextInt();
        switch (day) {
            case 1:
                System.out.println("Monday");
                break;
            case 2:
                System.out.println("Tuesday");
                break;
            case 3:
                System.out.println("Wednesday");
                break;
```

```
            case 4:
                System.out.println("Thursday");
                break;
            case 5:
                System.out.println("Friday");
                break;
            case 6:
                System.out.println("Saturday");
                break;
            case 7:
                System.out.println("Sunday");
                break;
        }
    }
}
```

Conditional Statements

WAP to check if the user entered year is leap or not

```
import java.util.*;
class Leap {
    public static void main(String[] args) {
        int yr;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the yr");
        yr=sc.nextInt();
        if((yr%4==0 && yr%100!=0) || (yr%100==0 && yr%400==0))
            System.out.println("Leap Year");
        else
            System.out.println("Not a Leap Year");
    } }
```

```
F:\Java>javac Leap.java
```

```
F:\Java>java Leap
Enter the yr
2024
Leap Year
```

```
F:\Java>java Leap
Enter the yr
2023
Not a Leap Year
```

Conditional Statements

- WAP to display the class according to the marks scored by student using switch case. The marks scored is taken input from the user and the class is displayed based on the range given below

Marks	Class
70-100	Distinction
60-69	First
50-59	Second
40-49	Pass
0-39	Fail

```
import java.util.*;
class Marks {
    public static void main(String[] args) {
        int marks;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter marks out of 100");
        marks=sc.nextInt();
        if (marks==100)
            System.out.println("Distinction");
        switch(marks/10)
        {
            case 0:
            case 1:
            case 2:
            case 3: System.out.println("Fail");
                    break;
            case 4: System.out.println("Pass Class");
                    break;
            case 5: System.out.println("Second Class");
                    break;
```

```
            case 6: System.out.println("First Class");
                    break;
            case 7:
            case 8:
            case 9: System.out.println("Distinction");
                    break;
            default: System.out.println("InvalidMarks");
        } } }
```

```
F:\Java>javac Marks.java

F:\Java>java Marks
Enter marks out of 100
85
Distinction

F:\Java>java Marks
Enter marks out of 100
37
Fail
```


Iterative Statements



For loop

```
import java.util.*;

class Natural{

    public static void main(String args[])
    {
        int i, n;

        Scanner sc=new Scanner
(System.in);

        System.out.println("Enter n : ");
        n=sc.nextInt();
        for(i=1;i<=n;i++)
        {
            System.out.println(i);
        }
    }
}
```

While loop

```
import java.util.*;

class NaturalWhile{

    public static void main(String args[])
    {
        int i, n;

        Scanner sc=new Scanner (System.in);
        System.out.println("Enter n : ");
        n=sc.nextInt();
        i=1;
        while(i<=n)
        {
            System.out.println(i);
            i++;
        }
    }
}
```

Do-while loop

```
import java.util.*;

class NaturalWhile{

    public static void main(String args[])
    {
        int i, n;

        Scanner sc=new Scanner
(System.in);

        System.out.println("Enter n : ");
        n=sc.nextInt();
        i=1;
        do
        {
            System.out.println(i);
            i++;
        }while(i<=n);
    }
}
```

Iterative Statements

WAP to find factorial of a number

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

```
F:\Java>java Fact
Enter the number
5
Factorial of 5 is 120
```

WAP to display the pattern

```
import java.util.*;
class StarPattern {
    public static void main(String[] args) {
        int n,i,j;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number");
        n=sc.nextInt();
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=i;j++)
            {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

```
F:\Java>javac StarPattern.java
F:\Java>java StarPattern
Enter the number
6
*
* *
* * *
* * * *
* * * * *
* * * * *
```

Arrays (single dimensional)

- ▶ an array is a collection of multiple data of same datatype
- ▶ Indexing in an array starts from 0. The last index is $n-1$ where n is the size of the array.
- ▶ Syntax for declaring array : `datatype array_name[]=new datatype[array_size];`
- ▶ Eg : `int arr[]=new int[10];`

Arrays (single dimensional)

WAP to input and display n elements in an array

```
import java.util.*;

class Array{

public static void main(String args[])

{

int i,n;

Scanner sc=new Scanner(System.in);

System.out.println("Enter no. of elements :");

n=sc.nextInt();

int a[]=new int[n];

for(i=0;i<=n-1;i++)

{

System.out.println("Enter a number :");

a[i]=sc.nextInt();

}
```

```
System.out.println("The elements in the array are :");

for(i=0;i<=n-1;i++)

{

System.out.print(a[i]+"\\t");

}}
```

```
F:\\Java>javac Array.java
```

```
F:\\Java>java Array
```

```
Enter no. of elements :
```

```
5
```

```
Enter a number :
```

```
1
```

```
Enter a number :
```

```
6
```

```
Enter a number :
```

```
7
```

```
Enter a number :
```

```
5
```

```
Enter a number :
```

```
4
```

```
The elements in the array are :
```

```
1      6      7      5      4
```

```
F:\\Java>
```

Arrays (single dimensional)

WAP to sort an array in ascending order

```
import java.util.*;

class SortedArray{

public static void main(String args[])

{

int i,n,temp;

Scanner sc=new Scanner(System.in);

System.out.println("Enter no. of elements :
");

n=sc.nextInt();

int a[]=new int[n];

for(i=0;i<=n-1;i++)

{

System.out.println("Enter a number :");

a[i]=sc.nextInt();

}
```

```
for(i=0;i<=n-2;i++)

{

for(int j=0;j<=n-2;j++)

{

if(a[j]>a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

System.out.println("Sorted Array: ");

for(i=0;i<=n-1;i++)

{

System.out.print(a[i]+"\\t");

}

}}
```

```
F:\Java>javac SortedArray.java
```

```
F:\Java>java SortedArray
```

```
Enter no. of elements :
```

```
5
```

```
Enter a number :
```

```
4
```

```
Enter a number :
```

```
9
```

```
Enter a number :
```

```
5
```

```
Enter a number :
```

```
2
```

```
Enter a number :
```

```
7
```

```
Sorted Array:
```

```
2
```

```
4
```

```
5
```

```
7
```

```
9
```

Arrays (multi dimensional)

- ▶ The Java multidimensional arrays are arranged as an array of arrays i.e. each element of a multi-dimensional array is another array.
- ▶ The representation of the elements is in rows and columns.
- ▶ Thus, you can get a total number of elements in a multidimensional array by multiplying row size with column size.
- ▶ So if you have a two-dimensional array of 3×4 , then the total number of elements in this array = $3 \times 4 = 12$
- ▶ The simplest of the multi-dimensional array is a two-dimensional array.
- ▶ A simple definition of 2D arrays is: A 2D array is an array of one-dimensional arrays.
- ▶ In Java, a two-dimensional array is stored in the form of rows and columns and is represented in the form of a matrix.
- ▶ The general declaration of a two-dimensional array is,
- ▶ `data_type[][] array_name = new data_type[row_size][column_size];`
- ▶ Eg : `int a[][]=new int[3][4];`

Arrays (multi dimensional)

```
import java.util.*;
class Array2d
{
    public static void main(String[] args) {
        int i,j,m,n;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter no. of rows and columns :
");
        m=sc.nextInt();
        n=sc.nextInt();
        int a[][] = new int[m][n];
        for (i=0;i<=m-1;i++) {
            for (j=0;j<=n-1;j++) {
                System.out.println("Enter a number : ");
                a[i][j]=sc.nextInt();
            }
        }
```

```
        System.out.println("The Array in matrix form is:");
        for (i=0;i<=m-1;i++) {
            for (j=0;j<=n-1;j++) {
                System.out.print(a[i][j]+"\\t");
            }
            System.out.println();
        }
    }
}
```

```
F:\Java>javac Array2d.java
F:\Java>java Array2d
Enter no. of rows and columns :
3
2
Enter a number :
1
Enter a number :
2
Enter a number :
3
Enter a number :
4
Enter a number :
5
Enter a number :
6
The Array in matrix form is:
1      2
3      4
5      6
```

Arrays (multi dimensional)



WAP to find and display sum of the diagonal elements of a square matrix

```
import java.util.*;
class Diagonal
{
    public static void main(String[] args) {
        int i,j,m,n,sum=0;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter no. of rows and columns :");
        m=sc.nextInt();
        n=sc.nextInt();
        int a[][] = new int[m][n];
        for (i=0;i<=m-1;i++) {
            for (j=0;j<=n-1;j++) {
                System.out.println("Enter a number :");
                a[i][j]=sc.nextInt();
            }
        }
```

```
        for (i=0;i<=m-1;i++) {
            for (j=0;j<=n-1;j++) {
                if(i==j)
                    sum=sum+a[i][j];
            }
        }
        System.out.println("Sum= "+sum);
    } }
```

```
F:\Java>javac SumofDiagonalElements.java

F:\Java>java Diagonal
Enter no. of rows and columns :
2
2
Enter a number :
4
Enter a number :
3
Enter a number :
9
Enter a number :
1
Sum= 5
```

Programming Practice Questions

- ▶ WAP to accept a number and display its equivalent ASCII character using type casting.
- ▶ WAP to find largest of three numbers accepted from the user through command line using ternary operator.
- ▶ WAP to display first n odd numbers
- ▶ WAP to calculate and display sum of square of first n natural numbers.
- ▶ WAP to display first n elements of Fibonacci series.
- ▶ WAP to calculate value of following series – $1 + 1/2 + 1/3 + 1/4 + \dots + 1/n$
- ▶ WAP to calculate value of following series – $1 + 1/2! + 1/3! + 1/4! + \dots + 1/n!$
- ▶ WAP to calculate and display value of s for $t=1,5,10,15,20,\dots,100$. Formula : $s = s_0 + v_0 + \frac{1}{2} a t^2$
- ▶ WAP to display following patterns

```
  *
 ***
 *****
 *******
*****
```

```
 *
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
      1
    1 2 1
  1 2 3 2 1
1 2 3 4 3 2 1
```

```
A
B C
D E F
G H I J
K L M N O
```

```
  *
 ***
*****
*****
*****
*****
***
 *
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Programming Practice Questions

- ▶ WAP to count number of digits in a user entered number using while loop.
- ▶ WAP to display a user entered number as a sequence of individual digits using while loop. Eg : 1691 should be displayed as 1 6 9 1
- ▶ WAP to check if a user entered number is divisible by 3 and 5 or not.
- ▶ WAP to check if the entered number is Armstrong number or not.
- ▶ WAP to display the class according to the marks scored by student using if else if. The marks scored is taken input from the user and the class is displayed based on the range given below

Marks	Class
70-100	Distinction
60-69	First
50-59	Second
40-49	Pass
0-39	Fail

- ▶ WAP to display a user digit number in words using switch case. Eg : i/p : 123 o/p : One Two Three
- ▶ WAP to find GCD and LCM of user entered two numbers.
- ▶ WAP to find largest element from the array.

Programming Practice Questions

- ▶ WAP to display average of n integers stored in an array
- ▶ WAP to search an element in an array and if found display the index.
- ▶ WAP to sort an array.
- ▶ WAP to add two matrices of size m x n
- ▶ WAP to find transpose of a matrix of size m x n
- ▶ The annual examination results of 5 students are tabulated as follows

Roll No	Subject 1	Subject 2	Subject 3
1			
2			
3			
4			
5			

WAP to read the data and determine the following

- Total marks obtained by each student
- The highest total marks and the student who obtained highest total marks.