

K. G. Engineering Institute

Bishnupur – Bankura – 722122

Department of Computer Science & Technology

3rd Year – 5th Semester

e-Contents : Java Programming - 531(S)

Chapter : Introduction to Java (Unit : 1)

Developed By :
Raj Kumar Datta (Lecturer & HOD)
Computer Science & Technology
KGEI, Bishnupur, Bankura- 722122

Contents



1 Introduction to Java

1.1 Fundamentals of Object Oriented Programming : Object and Classes, Data Abstraction and Encapsulation, Inheritance, Polymorphism, Dynamic Binding

1.2 Java Features : Compiled and Interpreted, Platform Independent and Portable, Object Oriented, Distributed, Multithreaded and Interactive, High Performance

1.3 Constant, Variable and Data Types, Constant Data Types, Scope of Variable, Symbolic Constant, Type Casting, Standard Default Values

1.4 Operator and Expression : Arithmetic, Relational, Logical, Assignment, Increment and decrement, Bitwise and Special Operators

1.5 Decision Making and Branching : Decision Making with if statement, Simple if statement, The if else statement, The else if ladder, The switch statement, The ? : Operator

1.6 Decision making and looping : The while statement, The do statement, The for statement, Jumps in Loops, Labeled Loops



1.1 Introduction to Java

❑ Fundamentals of Object Oriented Programming :

- ✓ Object-Oriented Programming (OOP) is the term used to describe a programming approach based on Object, Class, Encapsulation, Polymorphism, Inheritance, Data Abstraction.
- ✓ The object-oriented paradigm allows us to organize software as a collection of objects that consist of both data and function.
- ✓ Object-Oriented programming (OOP) refers to a type of programming in which programmers define the data type of a data structure and the type of operations that can be applied to the data structure.





1.1 Introduction to Java

□ Features of Object Oriented Programming :

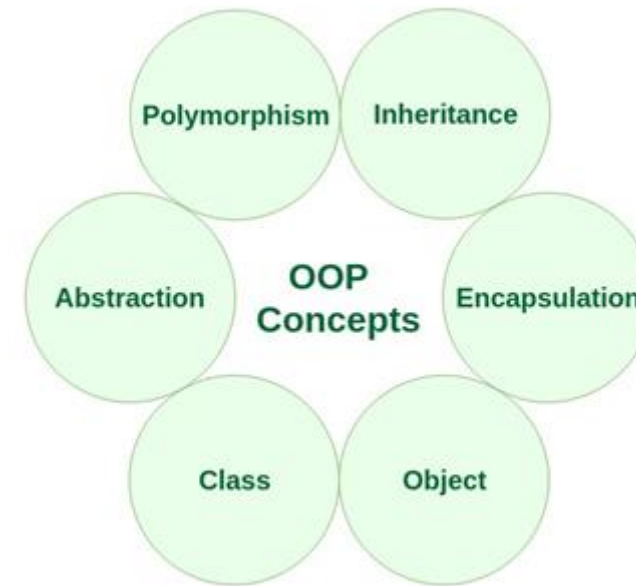
- ✓ Emphasis is on **data** rather than procedure.
- ✓ Programs are divided into **objects**.
- ✓ **Data structures** are designed such that they **characterize** the objects.
- ✓ Functions that **operate** on the data of an object are tied together.
- ✓ Data is **hidden** and **cannot be accessed** by external functions.
- ✓ Objects may communicate with **each other** through functions.
- ✓ New **data and functions** can easily be added **whenever necessary**.
- ✓ Follows **bottom-up approach**.



1.1 Introduction to Java

❑ Concepts of Object Oriented Programming :

- ✓ **Class**
- ✓ Object
- ✓ Encapsulation
- ✓ Polymorphism
- ✓ Inheritance
- ✓ Data Abstraction
- ✓ Message Passing





1.1 Introduction to Java : **Class**

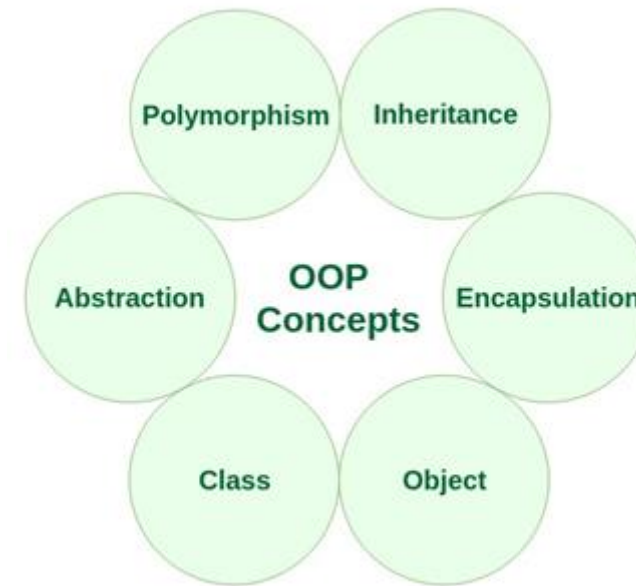
- ❑ A class is an user defined blueprint or prototype from which objects are created. It represents the set of properties that are common to all objects of one type.
- ❑ So, **Class** is a collection of data and function that manipulate the data. Another way, a class can be defined as a template / blueprint that describes the behavior / state that the object of its type support.
- ❑ **Class** can also be defined as thing which holds the properties of similar kind of Object.
- ❑ The **data components** of the class are called **properties** and the **function components** of the class are called **member methods**. The class that contains main function is called **main class**.



1.1 Introduction to Java

❑ Concepts of Object Oriented Programming :

- ✓ Class
- ✓ **Object**
- ✓ Encapsulation
- ✓ Polymorphism
- ✓ Inheritance
- ✓ Data Abstraction
- ✓ Message Passing





1.1 Introduction to Java : **Object**

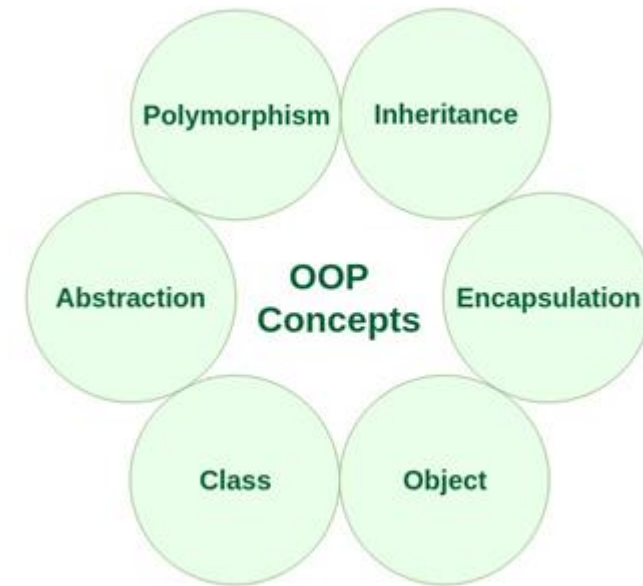
- ❑ An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance of a class.
- ❑ Definitions of an Object :
 - ✓ An object is *a real-world entity*.
 - ✓ An object is *a runtime entity*.
 - ✓ The object is *an entity which has state and behavior*.
 - ✓ The object is *an instance of a class*.
- ❑ An object has three characteristics :
 - ✓ **State** : represents the data (value) of an object.
 - ✓ **Behavior** : represents the behavior (functionality) of an object.
 - ✓ **Identity** : An object identity is typically implemented via unique ID. The value of the ID is not visible to the external user. However, JVM to identify each object uniquely.



1.1 Introduction to Java

❑ Concepts of Object Oriented Programming :

- ✓ Class
- ✓ Object
- ✓ **Encapsulation**
- ✓ Polymorphism
- ✓ Inheritance
- ✓ Data Abstraction
- ✓ Message Passing





1.1 Introduction to Java : **Encapsulation**

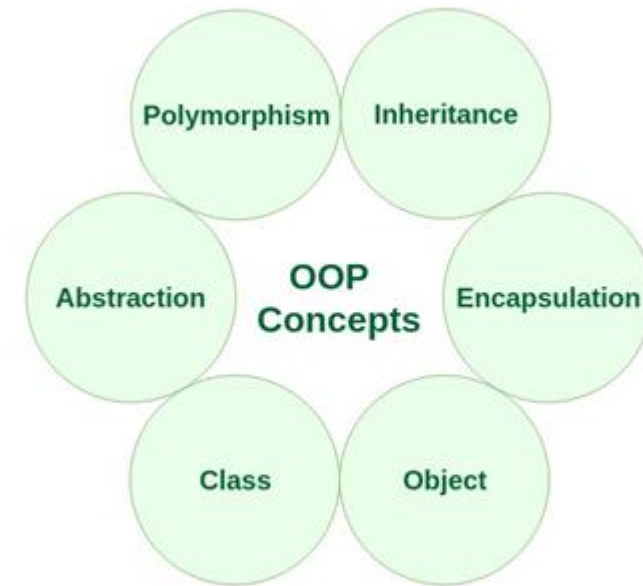
- ❑ The object-oriented paradigm encourages encapsulation. Encapsulation is used to hide the mechanics of the object, allowing the actual implementation of the object to be hidden, so that an user don't need to understand how the object works.
- ❑ **Encapsulation means to wrapping up of data and functions into a single unit. Here the data is not accessible outside of the class.**
- ❑ The data inside that class is accessible by the function in the same class. It is normally not accessible from the outside of the component. **i.e.** encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of it's own class in which they are declared.
- ❑ **As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding.**
- ❑ **Encapsulation can be achieved by declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.**



1.1 Introduction to Java

❑ Concepts of Object Oriented Programming :

- ✓ Class
- ✓ Object
- ✓ Encapsulation
- ✓ **Polymorphism**
- ✓ Inheritance
- ✓ Data Abstraction
- ✓ Message Passing





1.1 Introduction to Java : **Polymorphism**

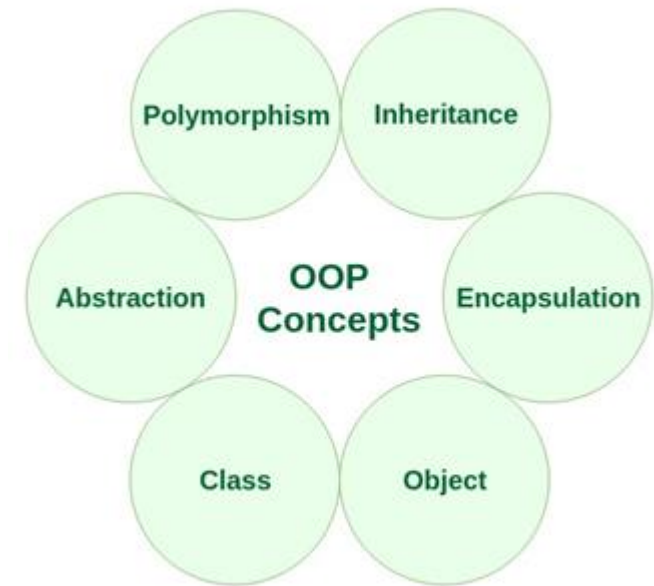
- ❑ Polymorphism means "multiple forms". In OOP these multiple forms refer to multiple forms of the same method, where the exact same method name can be used in different classes, or the same method name can be used in the same class with slightly different parameters.
- ❑ Polymorphism means the ability to take more than one form and refers to an operation exhibiting different behavior on instances.
- ❑ Object oriented programs use polymorphism to carry out the same operation in a manner customized to the object.
- ❑ There are two forms of polymorphism :
 - ✓ Method Overloading and
 - ✓ Method Overriding.



1.1 Introduction to Java

❑ Concepts of Object Oriented Programming :

- ✓ Class
- ✓ Object
- ✓ Encapsulation
- ✓ Polymorphism
- ✓ **Inheritance**
- ✓ Data Abstraction
- ✓ Message Passing





1.1 Introduction to Java : **Inheritance**

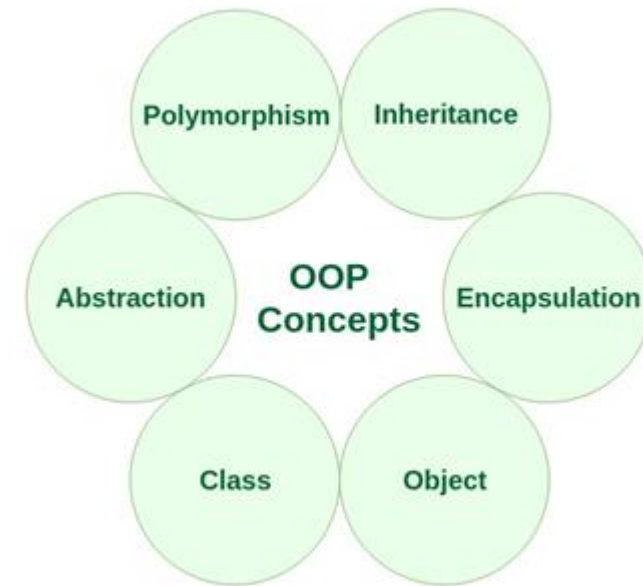
- ❑ Inheritance provides the idea of reusability.
- ❑ Inheritance is a mechanism of reusing the properties and / or extending the prosperities to the derive classes without modifying them, thus producing hierarchical relationships between them.
- ❑ Inheritance is a property by which the new classes are created using the old classes.
- ❑ The old classes are referred as base classes and the new classes are referred as derived classes. That means the derived classes inherit the properties of base class.
- ❑ In Inheritance, extends and implements keywords are used to describe in Java. The types of inheritances are available in Java are Single inheritance, Multi-level inheritance, Hierarchical inheritance, Hybrid inheritance.
- ❑ Java doesn't support Multiple Inheritance, rather support Multiple Interfaces.



1.1 Introduction to Java

❑ Concepts of Object Oriented Programming :

- ✓ Class
- ✓ Object
- ✓ Encapsulation
- ✓ Polymorphism
- ✓ Inheritance
- ✓ **Data Abstraction**
- ✓ Message Passing





1.1 Introduction to Java : **Abstraction**

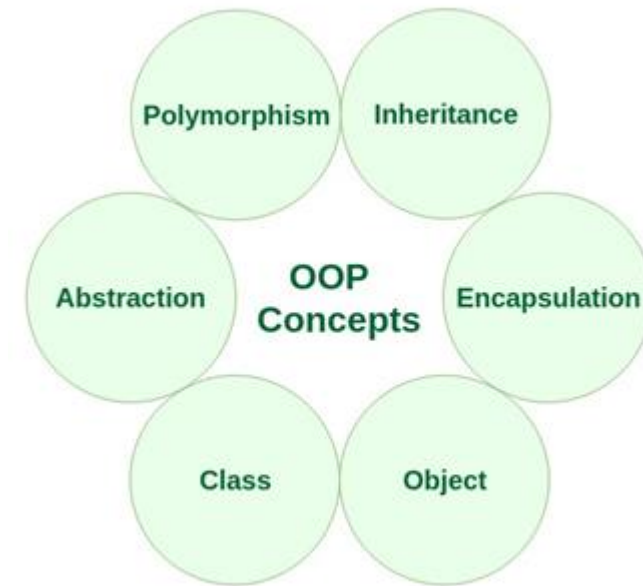
- ❑ Abstraction refers to the act of representing the essential features without including the background details or explanations. Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details.
- ❑ An abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user.
- ❑ In other words, the user will have the information on what the object does instead of how it does. The properties and behaviors of an object differentiate it from other objects of similar type and also help in classifying / grouping the objects.
- ❑ It reduces the complexity and increases the efficiency. Small program can easily be upgraded to large program. Software complexity can easily be managed.
- ❑ In java, abstraction is achieved by using Interfaces and Abstract classes.



1.1 Introduction to Java

❑ Concepts of Object Oriented Programming :

- ✓ Class
- ✓ Object
- ✓ Encapsulation
- ✓ Polymorphism
- ✓ Inheritance
- ✓ Data Abstraction
- ✓ **Message Passing**





1.1 Introduction to Java : **Message Passing**

- ☐ Object Oriented Programming also support message passing.
- ☐ The process of programming in which communication is involved is known as message passing.
- ☐ Message passing is the method of exchanging message between objects in Object Oriented Programming.
- ☐ It involves three basic steps. They are:
 - ✓ Creating classes
 - ✓ Creating objects
 - ✓ Establishing communication among objects.
- ☐ Message passing is also known as message exchanging.



Contents



2 Java Features

1.1 Fundamentals of Object Oriented Programming : Object and Classes, Data Abstraction and Encapsulation, Inheritance, Polymorphism, Dynamic Binding

1.2 Java Features : Compiled and Interpreted, Platform Independent and Portable, Object Oriented, Distributed, Multithreaded and Interactive, High Performance

1.3 Constant, Variable and Data Types, Constant Data Types, Scope of Variable, Symbolic Constant, Type Casting, Standard Default Values

1.4 Operator and Expression : Arithmetic, Relational, Logical, Assignment, Increment and decrement, Bitwise and Special Operators

1.5 Decision Making and Branching : Decision Making with if statement, Simple if statement, The if else statement, The else if ladder, The switch statement, The ? : Operator

1.6 Decision making and looping : The while statement, The do statement, The for statement, Jumps in Loops, Labeled Loops



1.2 Introduction to Java : Java Features

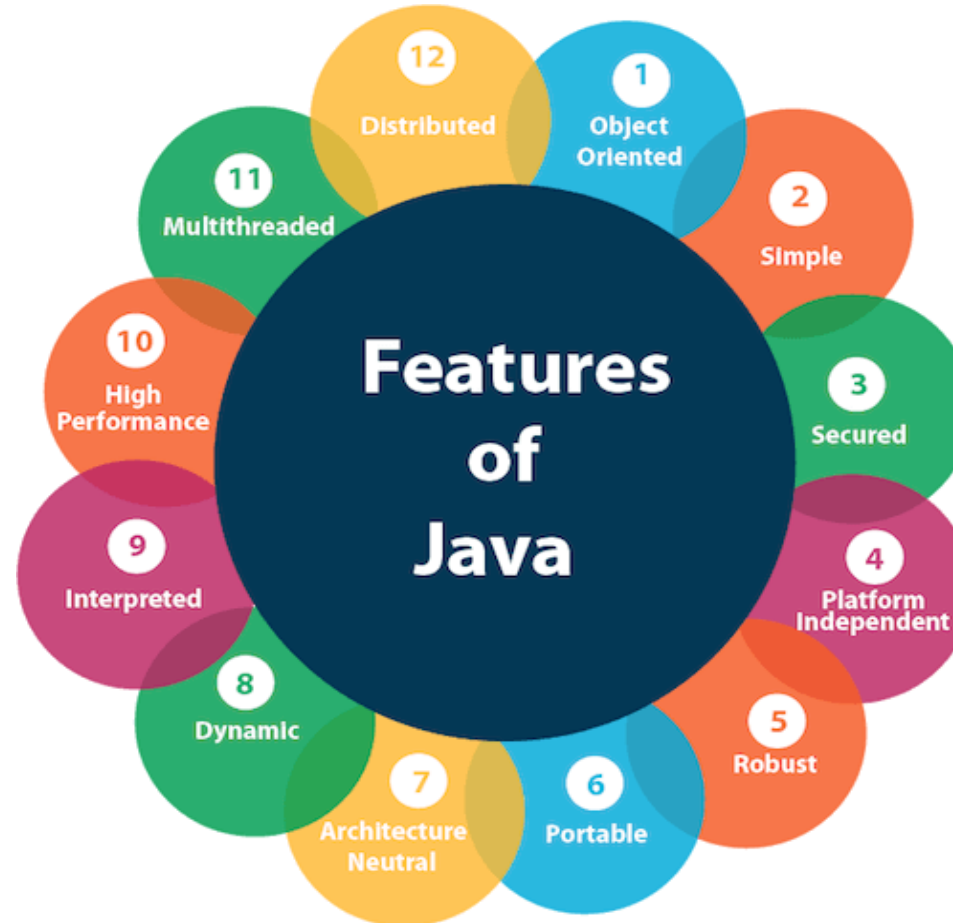
- ❑ JAVA was originated by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Seridan at **Sun Microsystem. Inc. in 1991 (Now , ORACLE JAVA)**.
- ❑ The language was initially named as “**OAK**” but replaced as “**JAVA**” in 1995.
- ❑ The primary motivation was the need for **Platform-Independent** i.e. **Architectural-Neutral language** that could be used to create software to be embedded in various electronic devices for remote access.
- ❑ JAVA is basically “**Internet version of C++**”. Though Java is neither upwardly or downwardly compatible with C++. Both of them were developed to solve the different set of problems.
- ❑ JAVA was basically developed for providing “**Write Once Run Anywhere**” access privilege typed application.



1.2 Introduction to Java : **Java Features**

☐ **JAVA Buzzwords :**

- ✓ Simple
- ✓ Secure
- ✓ Portable
- ✓ Robust
- ✓ Architectural Neutral
- ✓ Compiled and Interpreted



- ✓ Object Oriented
- ✓ Multithreaded
- ✓ Distributed
- ✓ Dynamic
- ✓ High Performance
- ✓ Platform Independence



1.2 Introduction to Java : Java Features

- ❑ **Simple** : Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun, Java language is a simple programming language because:
 - ✓ Java syntax is based on C++
 - ✓ Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading etc.
 - ✓ There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.
- ❑ **Secure** : Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:
 - ✓ No explicit pointer
 - ✓ Java Programs run inside a virtual machine (JVM)





1.2 Introduction to Java : **Java Features**

- ❑ **Portable** : The idea is that the Java language is portable or more precisely, the compiled byte code is portable. Following are the reasons :
 - ✓ Output of a Java compiler is **Non Executable Code i.e. Bytecode**.
 - ✓ Bytecode is a **highly optimized set of instructions**.
 - ✓ Bytecode is executed by Java run-time system, which is called the **Java Virtual Machine (JVM)**.
- ❑ **Robust** : Java is robust mainly for the following three reasons :
 - ✓ **Memory Management** : Java is very good in memory management. No use of pointers and no leakage of memory. **Garbage collection** is used for automatically to de-allocate the memory for unused objects.
 - ✓ **Exception handling and type checking mechanism** that is for run-time errors and
 - ✓ **Errors can be detected in the early stages.**





1.2 Introduction to Java : Java Features

- ❑ Architectural Neutral or Platform Independency : Java code is compiled into machine independent intermediate format i.e. bytecode, which can be executed on any systems for which Java Virtual Machine is supported. That means, Once writing a Java program, it will run irrespective of the Operating System without re-compiling. Thus, Java code is as “Write once, run anywhere”.
- ❑ Compiled and Interpreted : Java is a compiled programming language, but rather than compile straight to executable machine code, it compiles to an intermediate binary form called JVM byte code.
 - ✓ The byte code is then compiled and / or interpreted to run the program.
 - ✓ A Java interpreter or a just-in-time compiler (JIT) is used to run the compiled Java bytecode or machine independent code.





1.2 Introduction to Java : **Java Features**

- ❑ **Object Oriented** : JAVA is an Object Oriented Programming Language -
 - ❑ Firstly without declaring a class, no program can be written and compiled in Java. Java also supports **Class, Objects, Inheritance, Encapsulation, Polymorphism, Abstraction** and all user defined types are objects.
 - ❑ Java doesn't support these two properties :
 - ✓ All predefined types are objects (i.e. there is no difference between values which are objects and values which are primitive types.)
 - ✓ All operations performed on objects must be only through methods exposed at the objects.
 - ❑ **Static and Wrapper Classes** are also defying Java to be treated as **Purely or Completely Object Oriented Language**.
 - ❑ In Smalltalk, primitive values such as integer, boolean and character are also objects and Smalltalk is purely object oriented language.





1.2 Introduction to Java : Java Features

❑ Multithreaded : Java is a *multi-threaded programming language* which means multi-threaded program can be developed using Java. A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially utilizing multiple CPU time. That is, A single Java program can have many different threads executing independently and continuously.

❑ High Performance : To achieve high performance, Java relies on:

- ✓ Static optimizations done in the source-to-byte code compiler.
- ✓ Dynamic optimization done in the JIT compiler, based on run-time profiling.
- ✓ Sophisticated memory management algorithms that also reduces the workload.
- ✓ It is important in Java that, bad optimizations based on false assumptions can always be reassessed.





1.2 Introduction to Java : **Java Features**

- ❑ **Distributed** : Java is distributed because it encourages users to create distributed applications.
 - ✓ Distributed Programming is very helpful for developing large projects. Java helps programmer to achieve this by providing the concept of RMI (Remote Method Invocation) and EJB (Enterprise Java Beans).
 - ✓ Java supports networking and it's all extensive library of classes is used for interacting using TCP/IP and UDP protocols, which is much easier for creating network connections than other programming languages.
- ❑ **Dynamic** : Java gives the facility for dynamically linking new class libraries, packages, methods, and objects. It is highly dynamic as it can adapt to its evolving environment. This makes it easier for programmer to **expand** the classes and even **modify** them. This is achieved by extending the classes and / or implementing the interfaces.



1.2 Introduction to Java : **Java Technology**





1.2 Introduction to Java : **Java Technology**

- ❑ **Java technology is both a programming language and a platform.** The Java programming language is a high-level object-oriented language that has a particular syntax and style. A Java platform is a particular environment in which Java applications run.
- ❑ **Java Platform :** There are four platforms of the Java programming language:
 - ❑ **Standard Edition (Java SE - J2SE)**
 - ❑ **Micro Edition (Java ME - J2ME)**
 - ❑ **Enterprise Edition (Java EE – J2EE)**
 - ❑ **Java FX**
- ❑ All Java platforms consist of a Java Virtual Machine (VM) and an application programming interface (API).
- ❑ The Java Virtual Machine is a program, for a particular hardware and software platform, that runs Java technology applications.
- ❑ An API is a collection of software components that you can use to create other software components or applications. Each Java platform provides a virtual machine and an API, and this allows applications written for that platform to run on any compatible system.





1.2 Introduction to Java : **Java Technology**

- ❑ **Java SE** : When most people think of the Java programming language, they think of the Java SE API. Java SE's API provides the core functionality of the Java programming language. It defines everything from the basic types and objects of the Java programming language to high-level classes that are used for networking, security, database access, graphical user interface (GUI) development, and XML parsing.

In addition to the core API, the Java SE platform consists of a virtual machine, development tools, deployment technologies, and other class libraries and toolkits commonly used in Java technology applications.

- ❑ **Java EE** : The Java EE platform is built on top of the Java SE platform. The Java EE platform provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications.





1.2 Introduction to Java : **Java Technology**

- ❑ **Java ME** : The Java ME platform provides an API and a small-footprint virtual machine for running Java programming language applications on small devices, like mobile phones.

The API is a subset of the Java SE API, along with special class libraries useful for small device application development. Java ME applications are often clients of Java EE platform services.

- ❑ **Java FX** : JavaFX is a platform for creating rich internet applications using a lightweight user-interface API. JavaFX applications use hardware-accelerated graphics and media engines to take advantage of higher-performance clients and a modern look-and-feel as well as high-level APIs for connecting to networked data sources.

Java FX applications may be clients of Java EE platform services.





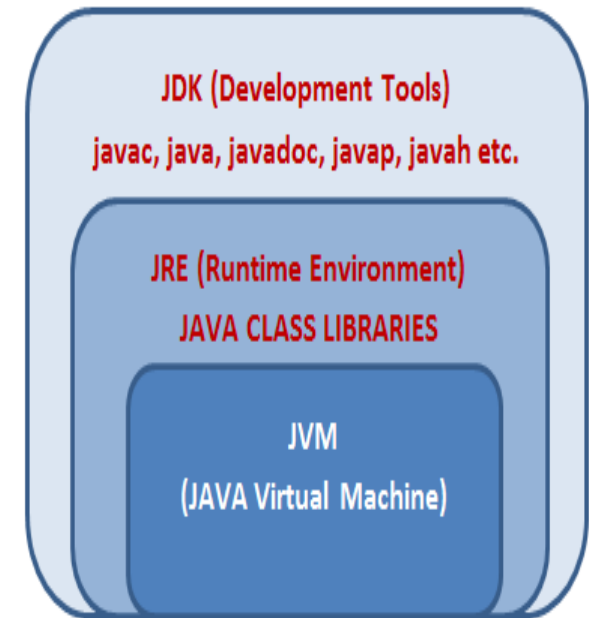
1.2 Introduction to Java : Java Technology

❑ Oracle has two products that implement Java Platform Standard Edition (Java SE) :

- ✓ Java SE Development Kit (JDK) and
- ✓ Java SE Runtime Environment (JRE).

❑ JDK is a superset of JRE , and contains everything that is in JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications.

❑ JRE provides the libraries, Java Virtual Machine, and other components to run applets and applications written in java.

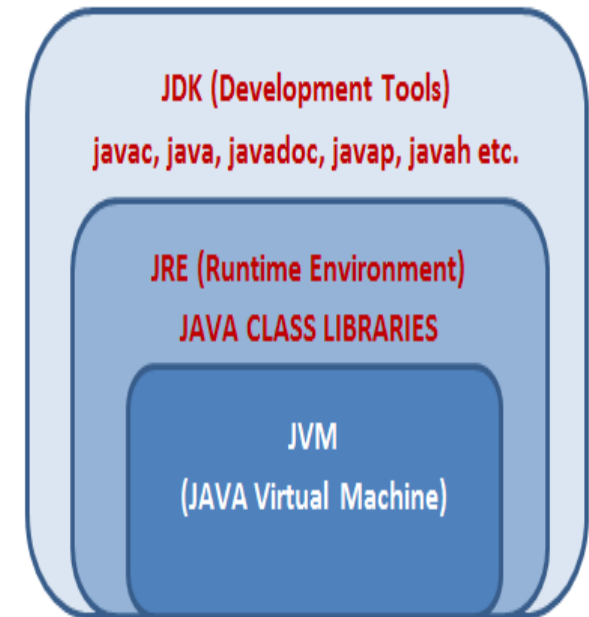




1.2 Introduction to Java : **Java Technology**

❑ Java JDK, JRE and JVM :

- ✓ **JVM (Java Virtual Machine)** is an abstract machine that enables the computer to run a Java program. At the time of running Java program, Java compiler first compiles Java code to bytecode. Then, JVM translates bytecode into native machine code.
- ✓ **JRE (Java Runtime Environment)** is a software package that provides Java class libraries, **Java Virtual Machine (JVM)**, and other components that are required to run Java applications.
- ✓ **JDK (Java Development Kit)** is a software development kit required to develop applications in Java. When JDK is downloaded, JRE is also downloaded with it. In addition to JRE, JDK also contains a number of development tools (compilers, JavaDoc, Java Debugger etc).



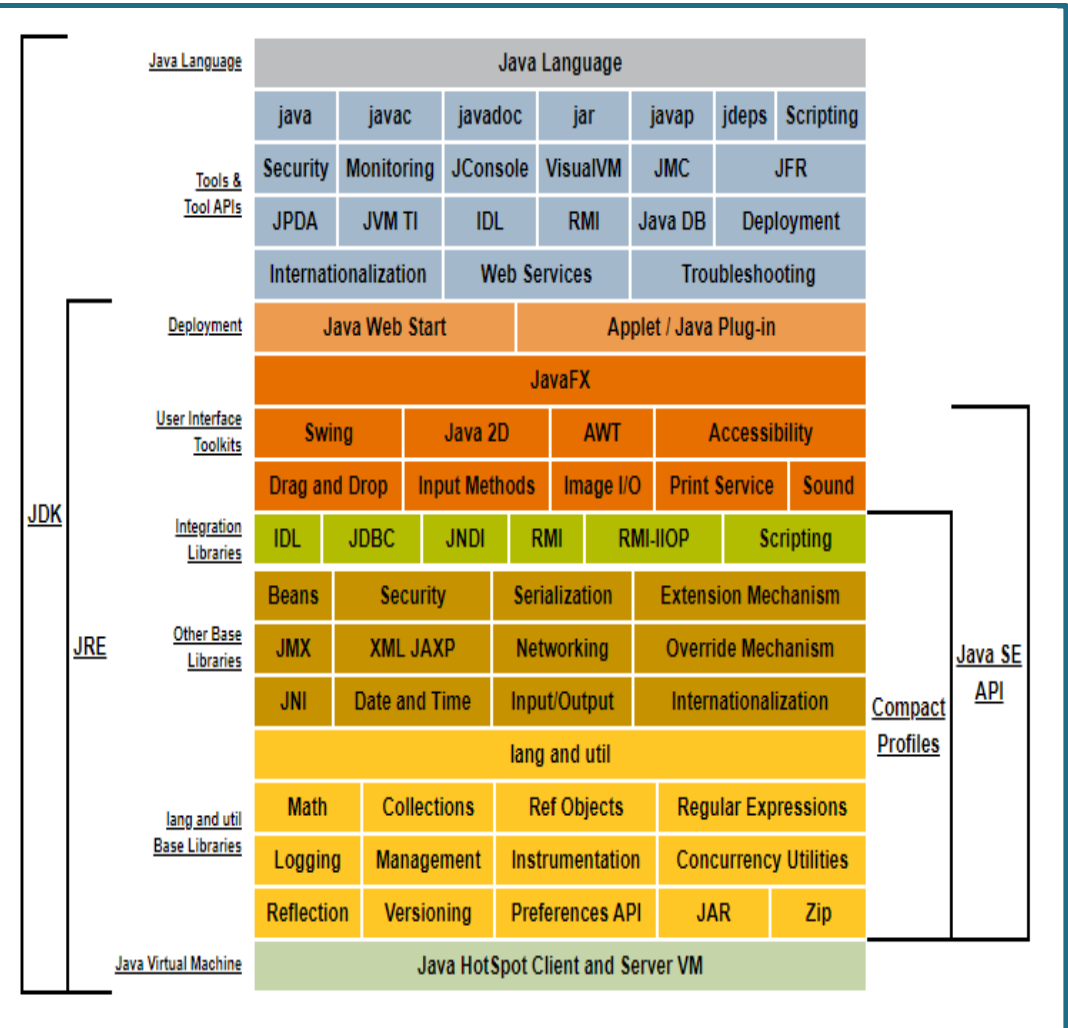


1.2 Introduction to Java : Java Technology

Java : the general-purpose, high-level programming language is a powerful software platform, gives the following features :

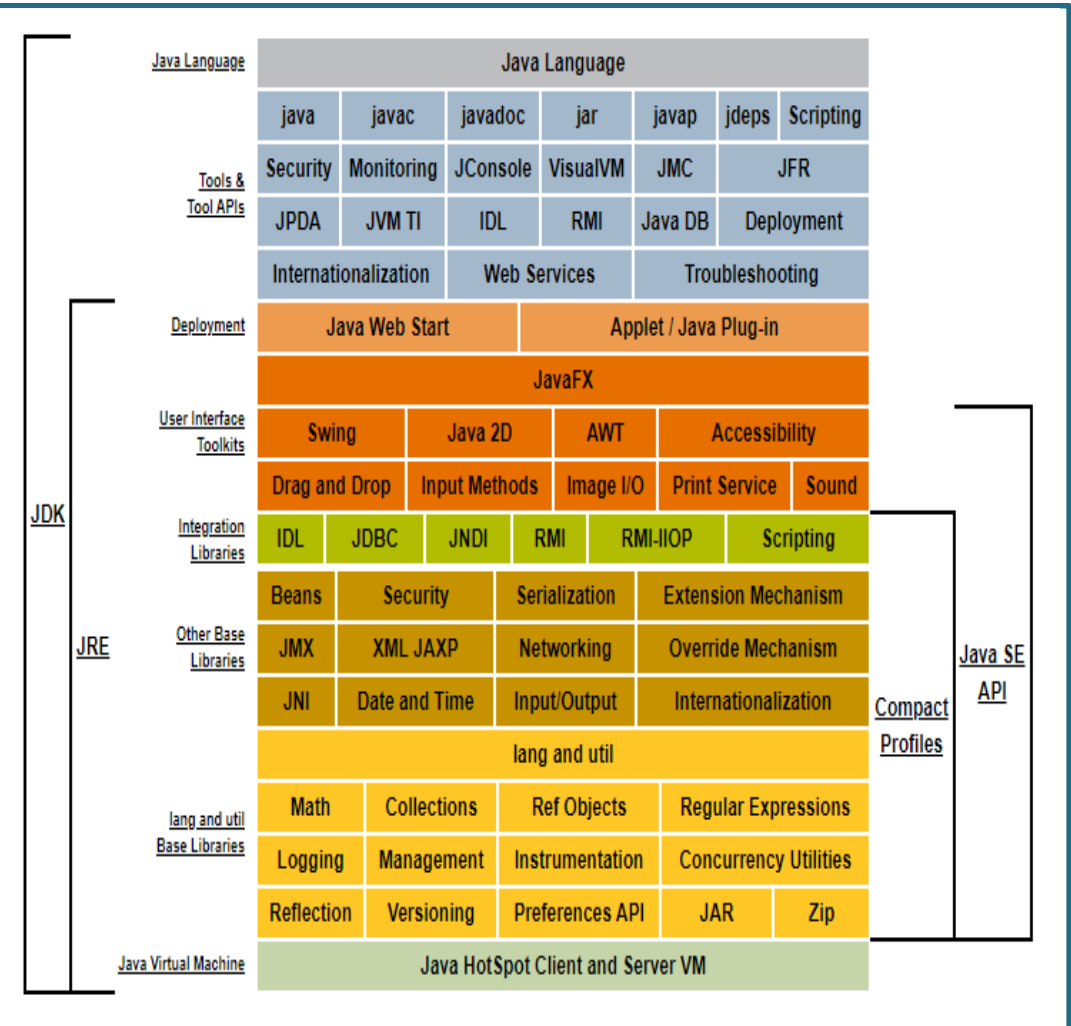
✓ **Development Tools** : The development tools provide everything need for compiling, running, monitoring, debugging, and documenting applications. The main tools that will be used for development are **javac compiler**, the **java launcher**.

✓ **Application Programming Interface (API)** : API provides the core functionality of the Java programming language. It offers a wide array of useful classes ready for use in any applications. It spans everything like basic objects, networking and security, XML generation and database access.



1.2 Introduction to Java : Java Technology

- ✓ **Deployment Technologies** : The JDK software provides standard mechanisms such as the Java Web Start software and Java Plug-In software for deploying applications to end users.
- ✓ **User Interface Toolkits** : The JavaFX, Swing, and Java 2D toolkits make it possible to create sophisticated Graphical User Interfaces (GUIs).
- ✓ **Integration Libraries** : Integration libraries such as Java IDL API, JDBC API, Java Naming and Directory Interface (JNDI) API, Java RMI, and Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology) enable database access and manipulation of remote objects.





1.2 Introduction to Java : **Java Technology**

❑ Java Tools :

| Tool Name | Brief Description |
|--------------|--|
| appletviewer | Run and debug applets without a web browser. |
| extcheck | Utility to detect Jar conflicts. |
| jar | Create and manage Java Archive (JAR) files. |
| java | The launcher for Java applications. A single launcher is used both for development and deployment. The old deployment launcher, JRE, is no longer provided. |
| javac | The compiler for the Java programming language. |
| javadoc | API documentation generator. |
| javah | C header and stub generator. Used to write native methods. |
| javap | Class file disassembler |
| jdb | The Java Debugger. |
| jdeps | Java class dependency analyzer |





1.2 Introduction to Java : Java Technology

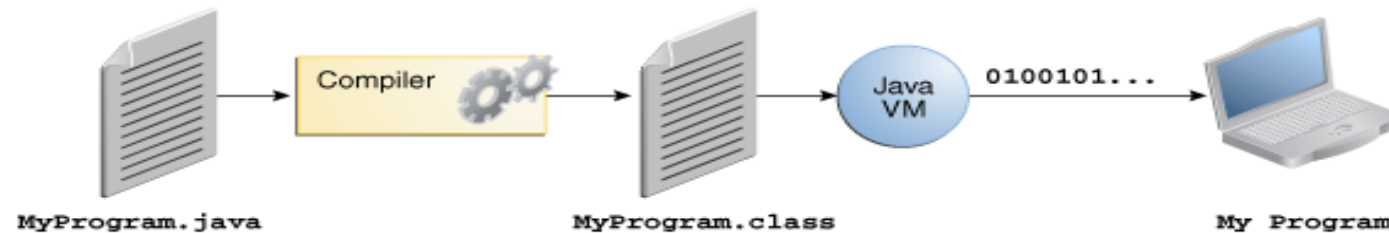
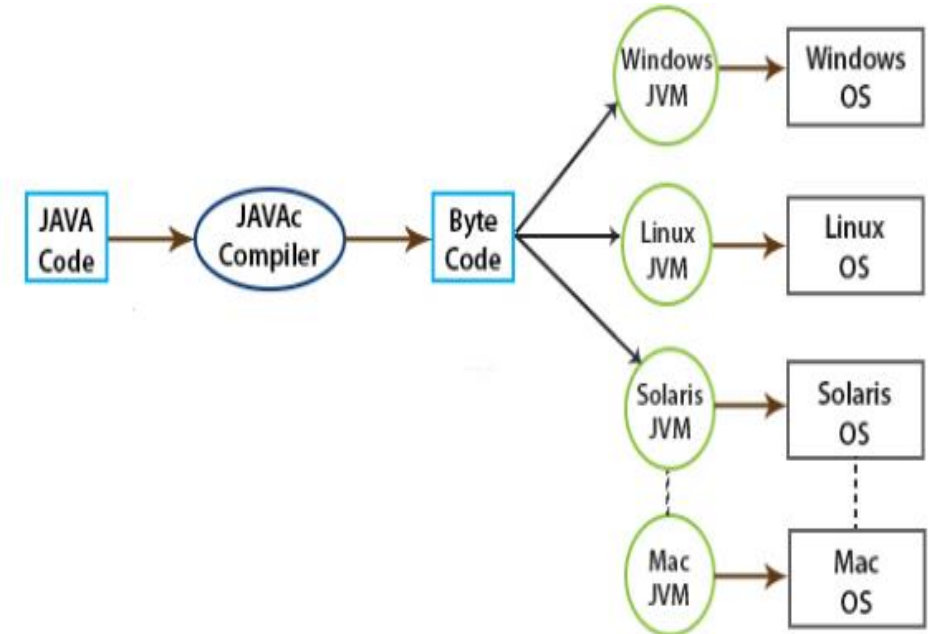
- ❑ Types of Application : Mainly 4 types of applications that can be created using Java programming:
 - ✓ Standalone Application : Standalone applications are also known as desktop applications or window-based applications. These are traditional software that need to be installed on every machine. Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.
 - ✓ 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.
 - ✓ Enterprise Application : An application that is distributed in nature, such as banking applications, etc. is called enterprise application. It has advantages of the high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.
 - ✓ 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.



1.2 Introduction to Java : Java Technology

❑ How to Compile and Run a JAVA Program :

- ✓ In the Java programming language, all **source code** is **first written in plain text** files ending with the **.java** extension.
- ✓ Those **source files** are then **compiled** into **.class** files by the **javac compiler**. A **.class** file does not contain code that is native to any processor; it instead contains *bytecode* - the machine language of the Java Virtual Machine (Java VM).
- ✓ The **java** launcher tool then runs the **java application** with an instance of the **Java Virtual Machine**.





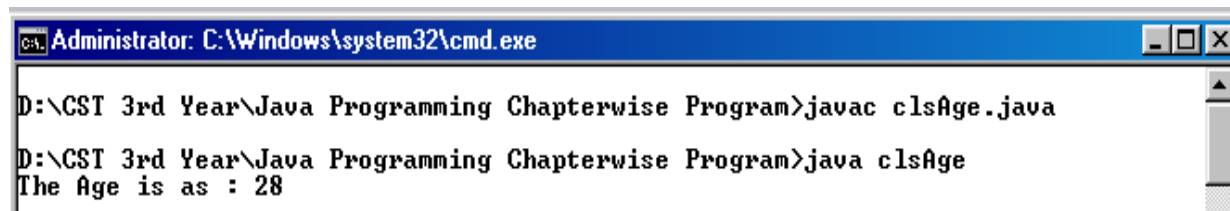
1.2 Introduction to Java : Java Technology

❑ Example for Saving, Compiling and Running a Program :

✓ First write a java program by the following way using any text editor (notepad).

✓ Code for Writing a Java program :

```
public class clsAge{  
    public static void main(String[] str){  
        int age = 28;  
        System.out.println("The Age is as : " + age);  
    }  
}
```



```
C:\Administrator: C:\Windows\system32\cmd.exe  
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge.java  
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge  
The Age is as : 28
```

- Save the file in any drive within a folder as “clsAge.java” i.e. every java file must be save with their extension as **<ClassName>.java**;
- Here it is named as clsAge.java, and this is the common practice to save the file name same as class name.
- Saving the Java file name can be different and at that time, running that java program, you have to interpret by the class name.





1.2 Introduction to Java : Java Technology

❑ Example for Saving, Compiling and Running a Program :

✓ First write a java program by the following way using any text editor (notepad).

✓ Code for Writing a Java program :

```
public class clsAge{  
    public static void main(String[] str){  
        int age = 28;  
        System.out.println("The Age is as : " + age);  
    }  
}
```

```
C:\Administrator: C:\Windows\system32\cmd.exe  
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge.java  
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge  
The Age is as : 28
```

- Java is secured and supports data hiding. So, every Java program must be written within a class.
- What is the meaning of public static void main(String[] str) ?
- Any java file will be accessed from outside the class. So, main method must be written with public access specifier.
- Without instantiating (i.e. Object), Java program can not called or execute. For that, static keyword is used.
- Return type of the main is void i.e. nothing will be returned from the main program.
- main() is a method compiler starts to execute from there and in main() in java only supports String type argument(s).





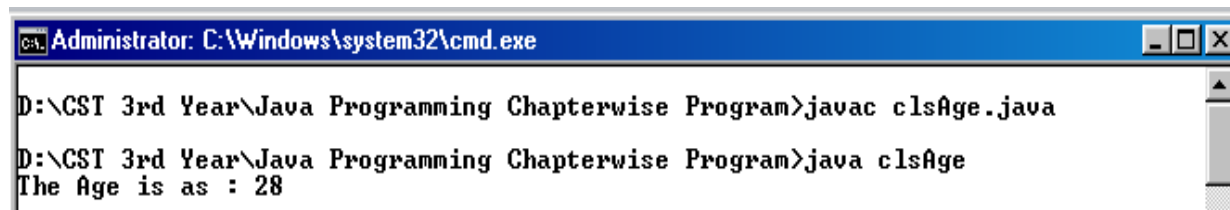
1.2 Introduction to Java : Java Technology

❑ Example for Saving, Compiling and Running a Program :

✓ First write a java program by the following way using any text editor (notepad).

✓ Code for Writing a Java program :

```
public class clsAge{  
    public static void main(String[] str){  
        int age = 28;  
        System.out.println("The Age is as : " + age);  
    }  
}
```



```
C:\Administrator: C:\Windows\system32\cmd.exe  
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge.java  
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge  
The Age is as : 28
```

- Java `System.out.println()` is used to print an argument that is passed to it. The statement can be broken into 3 parts which can be understood separately as:
- System: It is a final class defined in the java.lang package.
- out: This is an instance of PrintStream type, which is a public and static member field of the System class.
- println(): As all instances of PrintStream class have a public method `println()`, hence we can invoke the same on `out` as well. `System.out` represents the Standard Output Stream.



Contents



1 Introduction to Java

1.1 Fundamentals of Object Oriented Programming : Object and Classes, Data Abstraction and Encapsulation, Inheritance, Polymorphism, Dynamic Binding

1.2 Java Features : Compiled and Interpreted, Platform Independent and Portable, Object Oriented, Distributed, Multithreaded and Interactive, High Performance

1.3 Constant, Variable and Data Types, Constant Data Types, Scope of Variable, Symbolic Constant, Type Casting, Standard Default Values

1.4 Operator and Expression : Arithmetic, Relational, Logical, Assignment, Increment and decrement, Bitwise and Special Operators

1.5 Decision Making and Branching : Decision Making with if statement, Simple if statement, The if else statement, The else if ladder, The switch statement, The ? : Operator

1.6 Decision making and looping : The while statement, The do statement, The for statement, Jumps in Loops, Labeled Loops



1.3 Introduction to Java : Variable, Constant & Data Types

❑ **Variable** : The variable is the basic unit of storage in any programming language. A variable is defined by the combination of an *identifier*, a *type*, and an optional *initializer*. All variables have a *scope*, which defines their *visibility*, and a *lifetime*.

✓ **Syntax for declaring variable** :

type identifier1 [= value], [identifier2 [= value] ...];

here, *type* is one of Java's atomic types, or the name of a class or interface.

The identifier is the name of the variable and can be initialize the variable by specifying and equal sign and value.

❑ **Constant** : A constant is a variable whose value cannot be change once it has been assigned. Java doesn't have built-in support for constants. To define a variable as a constant, the keyword “final” is used in front of the variable declaration.

✓ **Syntax for declaring constant** :

final *type* identifier = value; // final float PI = 3.14159;





1.3 Introduction to Java : **Variable, Constant & Data Types**

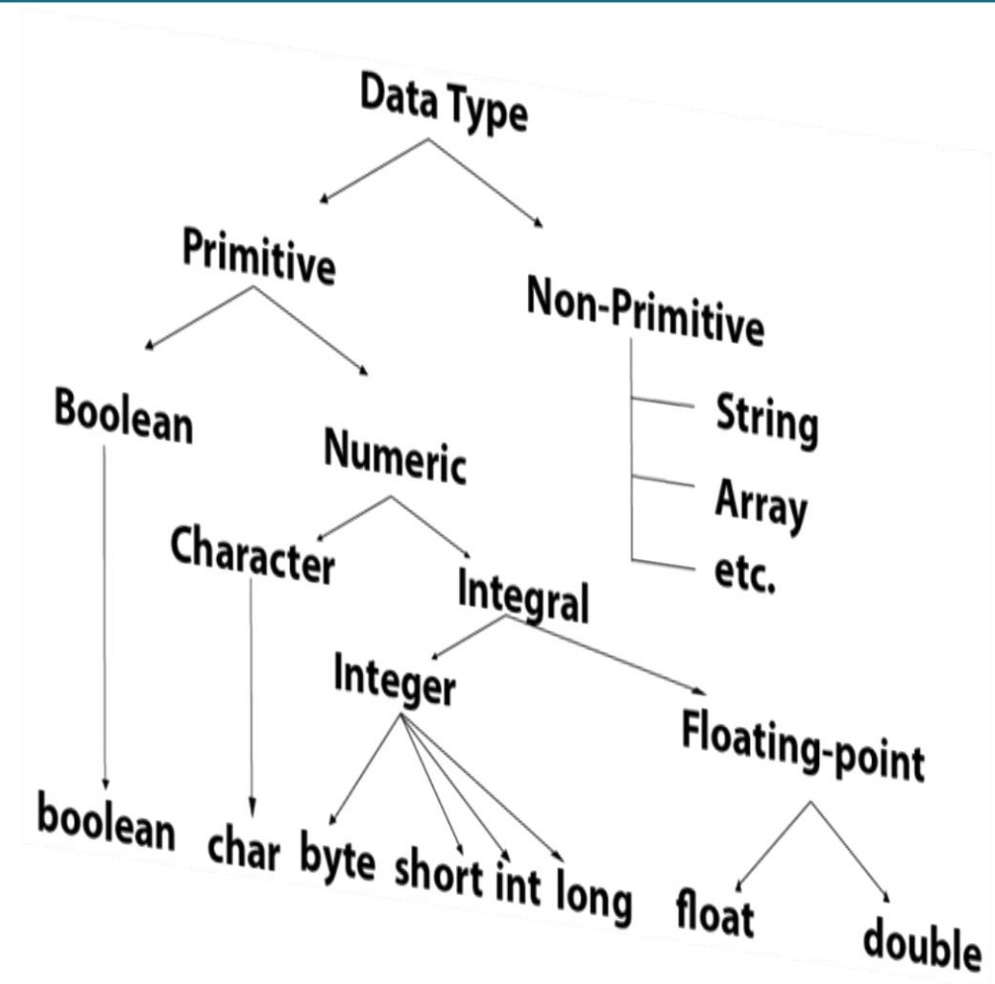
❑ **Data Type** : Data types specify the different sizes and values that can be stored in the variable. There are two types of data type in Java:

✓ **Primitive Data Type** :

The primitive data type includes boolean, char, byte, short, int, long, float and double.

✓ **Non-primitive Data Type** :

The non-primitive data type includes String and Arrays.





1.3 Introduction to Java : **Variable, Constant & Data Types**

- ❑ **Primitive Data Types** : The Java programming language is strictly-typed, which means that all variables must first be declared before they can be used. This involves stating the variable's type and name. A primitive type is predefined by the language and is named by a reserved keyword. Primitive values do not share state with other primitive values. The eight primitive data types supported by the Java programming language are:
 - ✓ **byte** : The byte data type is an 8-bit signed two's complement integer. It has a minimum value of -128 and a maximum value of 127. The byte data type can be useful for saving memory in large arrays, where the memory savings actually matters.
 - ✓ **short** : The short data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767. As with byte, this is similar and can also be use to save memory in large arrays, in situations where the memory savings actually matters.





1.3 Introduction to Java : **Variable, Constant & Data Types**

❑ Primitive Data Types :

- ✓ **int** : By default, the int data type is a 32-bit signed two's complement integer, which has a minimum value of -2^{31} and a maximum value of $2^{31}-1$. In Java SE 8 and later, the int data type is used to represent an unsigned 32-bit integer, which has a minimum value of 0 and a maximum value of $2^{32}-1$. Use the Integer class to use int data type as an unsigned integer.
- ✓ **long** : The long data type is a 64-bit two's complement integer. The signed long has a minimum value of -2^{63} and a maximum value of $2^{63}-1$. In Java SE 8 and later, it is used to represent an unsigned 64-bit long, which has a minimum value of 0 and a maximum value of $2^{64}-1$. Use this data type when you need a range of values wider than those provided by int.





1.3 Introduction to Java : **Variable, Constant & Data Types**

❑ Primitive Data Types :

- ✓ float : The float data type is a single-precision 32-bit IEEE 754 floating point. This data type should never be used for precise values, such as currency.
- ✓ double : The double data type is a double-precision 64-bit IEEE 754 floating point. For decimal values, this data type is generally the default choice. As mentioned above, this data type should never be used for precise values, such as currency.
- ✓ boolean : The boolean data type has only two possible values: true and false and used for simple flags that track boolean conditions. This data type represents one bit of information.
- ✓ char : The char data type is a single 16-bit Unicode character. It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65,535 inclusive).

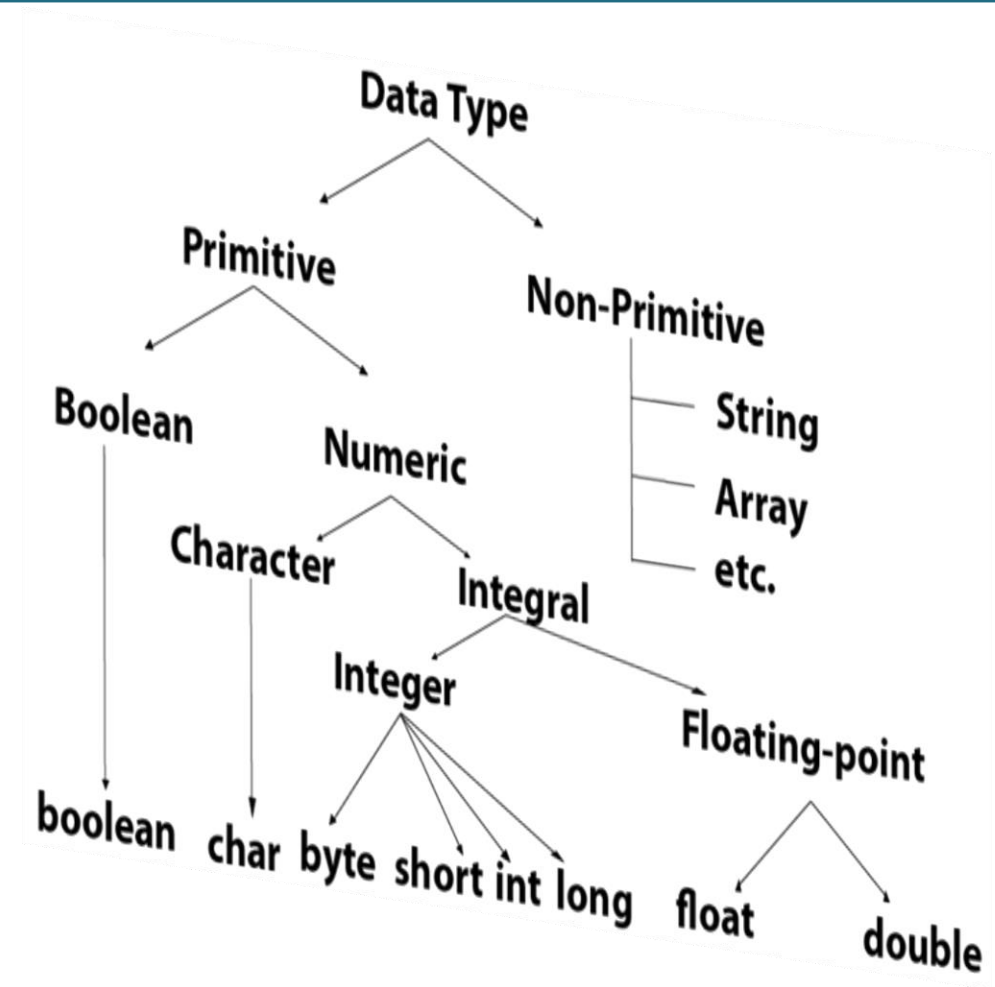




1.3 Introduction to Java : **Variable, Constant & Data Types**

❑ Size of each data type :

| Data Type | Default Size |
|-----------|--------------|
| boolean | 1 Byte |
| char | 2 Byte |
| byte | 1 Byte |
| short | 2 Byte |
| int | 4 Byte |
| long | 8 Byte |
| float | 4 Byte |
| double | 8 Byte |





1.3 Introduction to Java : **Variable, Constant & Data Types**

❑ **Non-primitive Data Type or Reference Data Type** : The Reference Data Type will contain a memory address of variable value because the reference types won't store the variable value directly in memory. They are **strings, arrays** etc.

✓ **String** : Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'. Below is the basic syntax for declaring a string in Java programming language.

✓ **Array** : An array is a group of like-typed variables that are referred to by a common name. Following are some important points about Java array.

- In Java all arrays are dynamically allocated.
- Since arrays are objects in Java, their length can be found by using member length.
- A Java array variable can also be declared like other variables with [] after the data type.
- The variables in the array are ordered and each have an index beginning from 0.
- Java array can also be used as a static field, a local variable or a method parameter.
- The **size** of an array must be specified by an int value and not long or short.





1.3 Introduction to Java : **Variable, Constant & Data Types**

- ❑ **Scope of Variable** : That means – the limit, as far as the variable can be used. In Java there are various types of variable scope:
 - ✓ **Local variables** : A variable that is declared within the method that is called local variables. It is defined in method or other statements, such as defined and used within any block, and outside the block or method, the variable cannot be used.
 - ✓ **Instance variables** : A non-static variable that is declared within the class but not in the method is called instance variable. Instance variables are related to a specific object; they can access class variables.
 - ✓ **Class / Static variables** : A variable that is declared with static keyword in a class but not in the method is called static or class variable.





1.3 Introduction to Java : **Variable, Constant & Data Types**

❑ **Symbolic Constant** : Symbolic constants are nothing more than a label or name that is used to represent a fixed value that never changes throughout a program.

- ✓ Final variables serve as symbolic constants.
- ✓ A final variable declaration is qualified with the reserved word **final**. The variable is set to a value in the declaration and cannot be reset. Any such attempt is caught at compile time.
- ✓ A variable can be declared as final using a keyword “final”. Doing so prevents its contents from being changed or modified in the program. This means that final variables are initialized where they are declared.
- ✓ **Example** : `final float PI = 3.14159;`





1.3 Introduction to Java : **Variable, Constant & Data Types**

❑ **Type Casting** : The process of converting the value of one data type (int, float, double, etc.) to another data type is known as typecasting. In Java, there are 13 types of type conversion. However, main 2 types are discussed herewith.

✓ **Widening Type Casting** : In the case of Widening Type Casting, the lower data type (having smaller size) is converted into the higher data type (having larger size). Hence there is no loss in data. This is why this type of conversion happens automatically. **This is also known as Implicit Type Casting.**

byte -> short -> char -> int -> long -> float -> double

✓ **Narrowing Type Casting** : In the case of Narrowing Type Casting, the higher data types (having larger size) are converted into lower data types (having smaller size). Hence there is the loss of data. This is why this type of conversion does not happen automatically. **This is also known as Explicit Type Casting.**

double -> float -> long -> int -> char -> short -> byte



Contents



1 Introduction to Java

1.1 Fundamentals of Object Oriented Programming : Object and Classes, Data Abstraction and Encapsulation, Inheritance, Polymorphism, Dynamic Binding

1.2 Java Features : Compiled and Interpreted, Platform Independent and Portable, Object Oriented, Distributed, Multithreaded and Interactive, High Performance

1.3 Constant, Variable and Data Types, Constant Data Types, Scope of Variable, Symbolic Constant, Type Casting, Standard Default Values

1.4 Operator and Expression : Arithmetic, Relational, Logical, Assignment, Increment and decrement, Bitwise and Special Operators

1.5 Decision Making and Branching : Decision Making with if statement, Simple if statement, The if else statement, The else if ladder, The switch statement, The ? : Operator

1.6 Decision making and looping : The while statement, The do statement, The for statement, Jumps in Loops, Labeled Loops





1.4 Introduction to Java : Operator & Expression

❑ Operator : An operator is a character that represents an action, for example + is an arithmetic operator that represents addition.

Types of Operator in Java :

- ✓ Arithmetic Operators
- ✓ Assignment Operators
- ✓ Auto-increment and Auto-decrement Operators
- ✓ Logical Operators
- ✓ Relational Operators
- ✓ Bitwise Operators
- ✓ Ternary Operator





1.4 Introduction to Java : Operator & Expression

❑ Arithmetic Operators : Operators are used to perform mathematical operations like addition, subtraction, etc. Assume that $A=20$ and $B=30$ for the below table.

| Operator | Description | Example |
|------------------|---|---------------|
| + Addition | Adds values on either side of the operator | $A + B = 50$ |
| – Subtraction | Subtracts the right-hand operator with left-hand operator | $A - B = -10$ |
| * Multiplication | Multiplies values on either side of the operator | $A * B = 600$ |
| / Division | Divides left hand operand with right hand operator | $A / B = 0$ |
| % Modulus | Divides left hand operand by right hand operand and returns remainder | $A \% B = 20$ |





1.4 Introduction to Java : Operator & Expression

❑ Operator : An operator is a character that represents an action, for example + is an arithmetic operator that represents addition.

Types of Operator in Java :

- ✓ Arithmetic Operators
- ✓ Assignment Operators
- ✓ Auto-increment and Auto-decrement Operators
- ✓ Logical Operators
- ✓ Relational Operators
- ✓ Bitwise Operators
- ✓ Ternary Operator





1.4 Introduction to Java : Operator & Expression

□ Assignment Operators : An Assignment Operator is an operator used to assign a new value to a variable.

| Operator | Description | Example |
|------------|--|----------------|
| = | Assigns values from right side operands to left side operand | C = A + B |
| += | It adds right operand to the left operand and assigns the result to left operand | C += A |
| -= | It subtracts right operand from the left operand and assigns the result to left operand | C -= A |
| *= | It multiplies right operand with the left operand and assigns the result to left operand | C *= A |
| /= | It divides left operand with the right operand and assigns the result to left operand | C /= A |
| %= | It takes modulus using two operands and assigns the result to left operand | C %= A |
| \wedge = | Performs exponential (power) calculation on operators and assign value to the left operand | C \wedge = A |





1.4 Introduction to Java : Operator & Expression

❑ Operator : An operator is a character that represents an action, for example + is an arithmetic operator that represents addition.

Types of Operator in Java :

- ✓ Arithmetic Operators
- ✓ Assignment Operators
- ✓ Auto-increment and Auto-decrement Operators
- ✓ Logical Operators
- ✓ Relational Operators
- ✓ Bitwise Operators
- ✓ Ternary Operator





1.4 Introduction to Java : Operator & Expression

❑ Auto-increment and Auto-decrement Operators : ++ and – operators are used for prefix or postfix increment or decrement operations in Java.

| Operator | Description | Example |
|--------------|---|---------|
| ++ (prefix) | prefix Increment Operator: increments value by 1 | ++A; |
| (postfix) ++ | postfix Increment Operator: increments value by 1 | A++; |
| -- (prefix) | prefix Decrement Operator: decrements value by 1 | --B; |
| (postfix) -- | postfix Decrement Operator: decrements value by 1 | B--; |





1.4 Introduction to Java : Operator & Expression

❑ Operator : An operator is a character that represents an action, for example + is an arithmetic operator that represents addition.

Types of Operator in Java :

- ✓ Arithmetic Operators
- ✓ Assignment Operators
- ✓ Auto-increment and Auto-decrement Operators
- ✓ Logical Operators
- ✓ Relational Operators
- ✓ Bitwise Operators
- ✓ Ternary Operator





1.4 Introduction to Java : Operator & Expression

❑ Logical Operators : Logical Operators are used with binary variables. They are mainly used in conditional statements and loops for evaluating a condition.

| Operator | Description | Example |
|----------|--|--------------------------------|
| && | A && B will return true if both the variables are true else return false. | TRUE = TRUE && TRUE |
| | A B will return false if both the variables are false else return true. | FALSE = FALSE FALSE |
| ! | !A would return the opposite of A, that means it would be true if A is false and it would return false if A is true. | TRUE = !FALSE FALSE = !TRUE |





1.4 Introduction to Java : Operator & Expression

❑ Operator : An operator is a character that represents an action, for example $+$ is an arithmetic operator that represents addition.

Types of Operator in Java :

- ✓ Arithmetic Operators
- ✓ Assignment Operators
- ✓ Auto-increment and Auto-decrement Operators
- ✓ Logical Operators
- ✓ Relational Operators
- ✓ Bitwise Operators
- ✓ Ternary Operator





1.4 Introduction to Java : Operator & Expression

❑ Relational Operators : These operators compare the values on either side of them and decide the relation among them. Assume A = 10 and B = 20 for the below table.

| Operator | Description | Example |
|----------|---|----------------------|
| == | If the values of two operands are equal, then the condition becomes true. | (A == B) is not true |
| != | If the values of two operands are not equal, then condition becomes true. | (A != B) is true |
| > | If the value of the left operand is greater than the value of right operand, then condition becomes true. | (A > B) is not true |
| < | If the value of the left operand is less than the value of right operand, then condition becomes true. | (A < B) is true |
| >= | If the value of the left operand is greater than or equal to the value of the right operand, then condition becomes true. | (A >= B) is not true |
| <= | If the value of the left operand is less than or equal to the value of right operand, then condition becomes true. | (A <= B) is true |





1.4 Introduction to Java : Operator & Expression

❑ Operator : An operator is a character that represents an action, for example + is an arithmetic operator that represents addition.

Types of Operator in Java :

- ✓ Arithmetic Operators
- ✓ Assignment Operators
- ✓ Auto-increment and Auto-decrement Operators
- ✓ Logical Operators
- ✓ Relational Operators
- ✓ Bitwise Operators
- ✓ Ternary Operator





1.4 Introduction to Java : Operator & Expression

❑ **Bitwise Operators** : Bitwise operations directly manipulate **bits**. In all Systems, numbers are represented with bits i.e. series of zeros and ones. In fact, everything in a computer is represented by bits. Assume that $A = 4(0000\ 0100)$ and $B = 8(0000\ 1000)$ for the below table.

| Operator | Description | Example |
|----------------|-------------------------------|--------------|
| & (AND) | Shows bit by bit AND of input | $A \& B$ |
| (OR) | Shows OR of input | $A B$ |
| ^ (XOR) | Shows XOR of input | $A \wedge B$ |
| ~ (Complement) | Shows the one's complement. | $\sim A$ |





1.4 Introduction to Java : Operator & Expression

❑ Operator : An operator is a character that represents an action, for example $+$ is an arithmetic operator that represents addition.

Types of Operator in Java :

- ✓ Arithmetic Operators
- ✓ Assignment Operators
- ✓ Auto-increment and Auto-decrement Operators
- ✓ Logical Operators
- ✓ Relational Operators
- ✓ Bitwise Operators
- ✓ Ternary Operator





1.4 Introduction to Java : Operator & Expression

❑ Ternary Operator : This operator evaluates a boolean expression. The conditional operator or ternary operator `?:` is shorthand for the if-then-else statement.

❑ **Syntax** : `datatype variable = (Condition) ? Statement 1 : Statement 2`

Condition : It is the expression to be evaluated which returns a boolean value.

Statement 1 : It is the statement to be executed if the condition results in a true state.

Statement 2 : It is the statement to be executed if the condition results in a false state.

❑ **Example** : `int MaxAge = (Student1.Age > Student2.Age) ? Student1.Age : Student2.Age`

i.e. if (the age of Student1 is greater than the age of Student2 then MaxAge will be initialized as Student1.Age else Student2.Age.



Contents



1 Introduction to Java

1.1 Fundamentals of Object Oriented Programming : Object and Classes, Data Abstraction and Encapsulation, Inheritance, Polymorphism, Dynamic Binding

1.2 Java Features : Compiled and Interpreted, Platform Independent and Portable, Object Oriented, Distributed, Multithreaded and Interactive, High Performance

1.3 Constant, Variable and Data Types, Constant Data Types, Scope of Variable, Symbolic Constant, Type Casting, Standard Default Values

1.4 Operator and Expression : Arithmetic, Relational, Logical, Assignment, Increment and decrement, Bitwise and Special Operators

1.5 Decision Making and Branching : Decision Making with if statement, Simple if statement, The if else statement, The else if ladder, The switch statement, The ? : Operator

1.6 Decision making and looping : The while statement, The do statement, The for statement, Jumps in Loops, Labeled Loops

1.5 Introduction to Java : Decision Making & Branching



❑ Decision Making :

- ✓ Decision making statement used to control the flow of execution of program depending upon condition, means if condition is true then the block will execute and if condition is false then block will not execute.

✓ **Types Of Decision Making Statement :**

In java there are four types of decision making statement, and they are mentioned below :

- if statement(s)
- if-else statement(s)
- Ladder if-else statement(s)
- switch statement
- The ? : operator (**Ternary Operator / Shorthand type of if-else statement(s)**)





1.5 Introduction to Java : Decision Making & Branching

❑ Decision making with if statement :

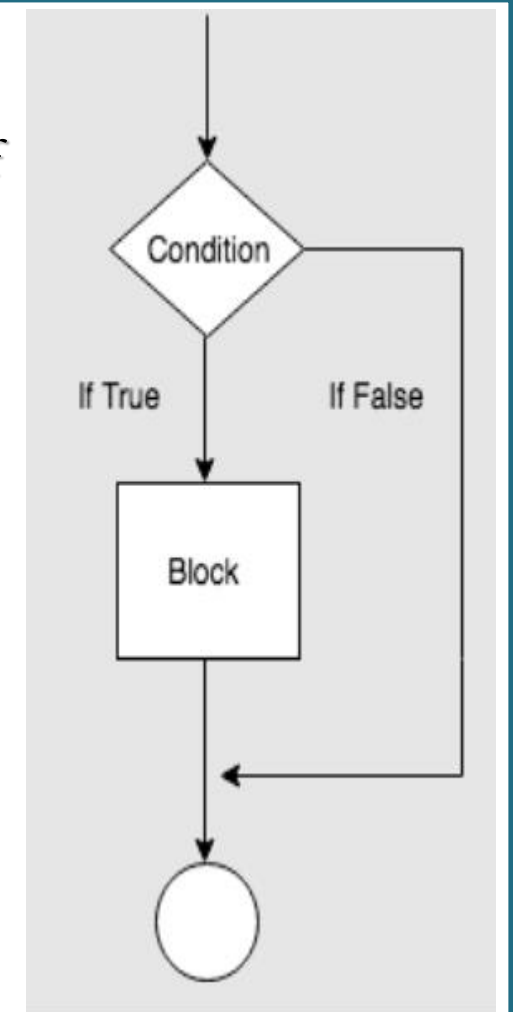
- ✓ if the condition satisfied then the code block will execute otherwise code block will not execute. i.e. If statement will execute block of statements only if the condition is true.

✓ **Syntax :**

```
if(condition){    Block of statement(s)    }
```

✓ **Example :**

```
public class clsAgeI {  
    public static void main(String[] str){  
        int age = 28;  
        if (age == 28){  
            System.out.println("Age is 28");  
        }  
    }  
}
```





1.5 Introduction to Java : Decision Making & Branching

❑ Decision making with if - else statement :

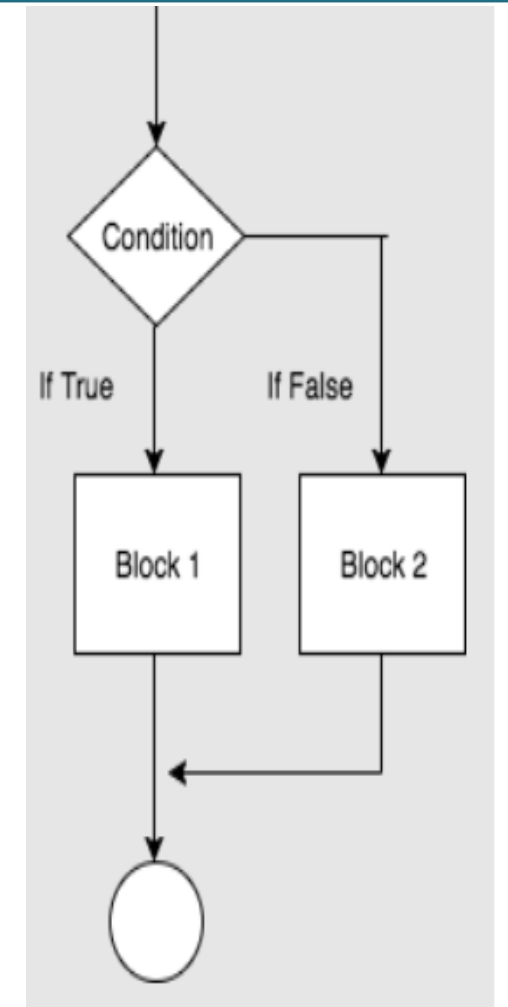
- ✓ If else statements will execute one block of statements if condition is true else another block of statement will be executed.

- ✓ **Syntax :**

```
if(condition){ Statement1(s)
               }else{ statement2(s)}
```

- ✓ **Example :**

```
public class clsAge2{
    public static void main(String[] str){
        int age = 28;
        if (age == 28){ System.out.println("Age is
28");
        }else{
            System.out.println("Age is different.");
        }
    }
}
```





1.5 Introduction to Java : Decision Making & Branching

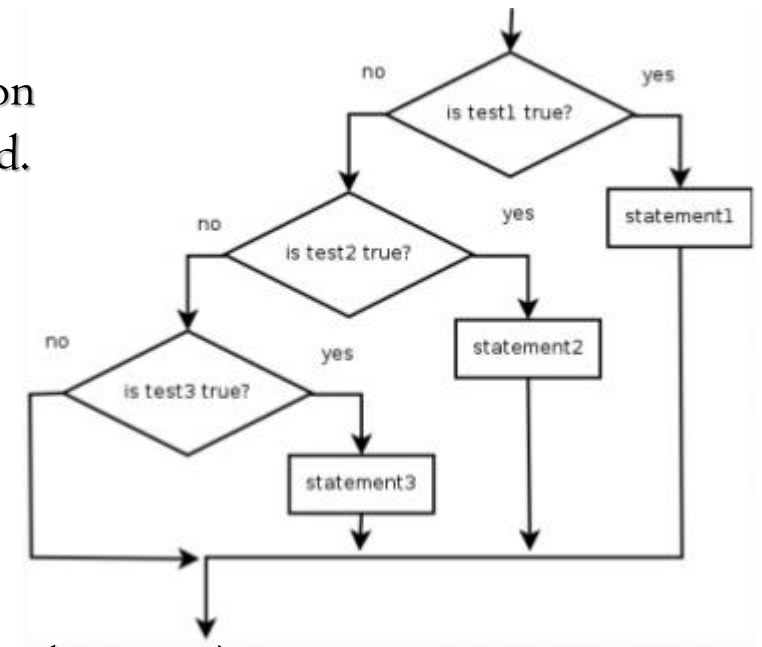
❑ Decision making with else if ladder statement :

- ✓ If else statements will execute one block of statements if condition is true else another if – else block of statement(s) will be executed.

✓ **Syntax :** if(condition){ Statement1(s)}
 else if(condition){ Statement2(s)}
 else{ Statement3(s)}

✓ **Example :**

```
public class clsAge3{  
    public static void main(String[] str){  
        int age = 28;  
        if (age > 28){System.out.println("Age is greater than 28.");  
        }else if (age == 28){ System.out.println("Age is 28");  
        }else{       System.out.println("Age is lesser than 28.");  
        }  
    }  
}
```



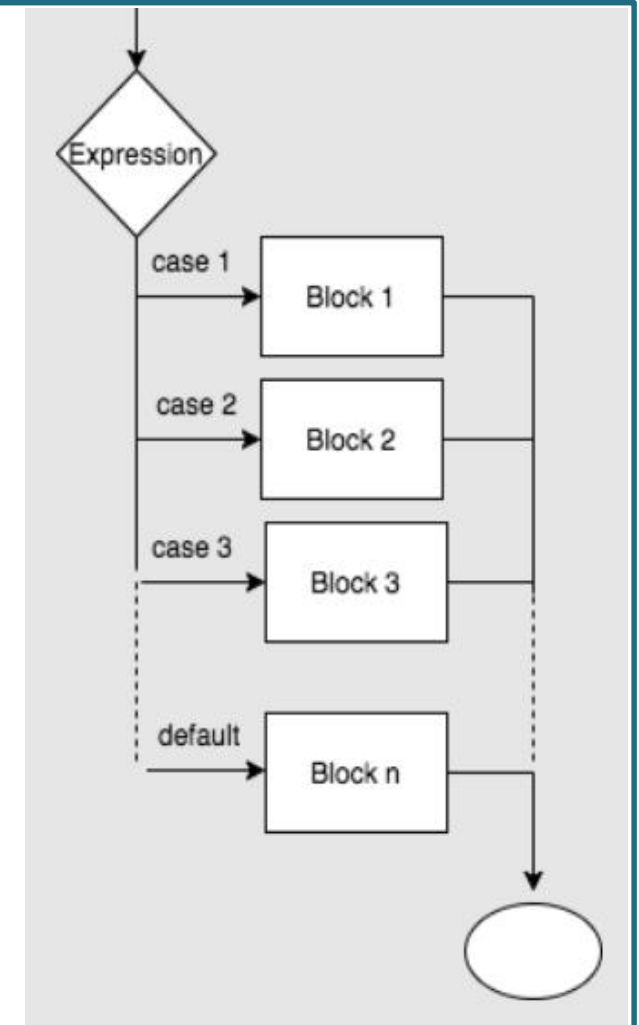
1.5 Introduction to Java : Decision Making & Branching

❑ Decision making with switch statement :

- ✓ Switch statement in Java compare a variable with list of values. It is like a nested if else but better alternative than large series of nested if else statements.

- ✓ **Syntax :**

```
switch(expression) {  
    case value : Statement(s);  
                break;  
    case value : Statements(s);  
                break;  
    default   :  
                Statement(s);  
}
```

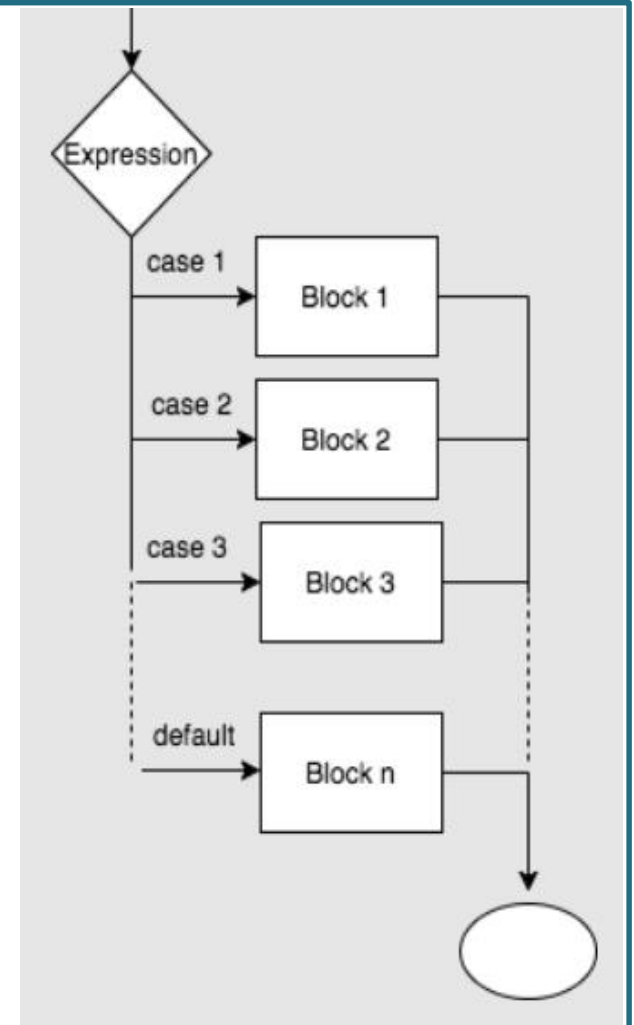


1.5 Introduction to Java : Decision Making & Branching

❑ Decision making with switch statement :

✓ **Example :**

```
public class clsAge4{  
    public static void main(String[] str){  
        char grade = 'D';  
        switch(grade) {  
            case 'A': System.out.println("Excellent!"); break;  
            case 'B': System.out.println("Very Good"); break;  
            case 'C': System.out.println("Good"); break;  
            case 'D': System.out.println("Fail"); break;  
            default : System.out.println("Invalid Grade.");  
        }  
    }  
}
```





1.5 Introduction to Java : Decision Making & Branching

❑ Decision making for the ?: operator :

```
public class clsAge5{  
    public static void main(String[] str){  
        int MaxAge, age1, age2;  
        age1 = Integer.parseInt(str[0]);  
        age2 = Integer.parseInt(str[1]);  
        MaxAge = (age1 > age2) ? age1 : age2;  
        System.out.println("The maximum age is as : " + MaxAge);  
    }  
}
```

```
Administrator: C:\Windows\system32\cmd.exe  
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge5.java  
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge5 12 34  
The maximum age is as : 34  
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge5 34 12  
The maximum age is as : 34  
D:\CST 3rd Year\Java Programming Chapterwise Program>_
```



Contents



1 Introduction to Java

1.1 Fundamentals of Object Oriented Programming : Object and Classes, Data Abstraction and Encapsulation, Inheritance, Polymorphism, Dynamic Binding

1.2 Java Features : Compiled and Interpreted, Platform Independent and Portable, Object Oriented, Distributed, Multithreaded and Interactive, High Performance

1.3 Constant, Variable and Data Types, Constant Data Types, Scope of Variable, Symbolic Constant, Type Casting, Standard Default Values

1.4 Operator and Expression : Arithmetic, Relational, Logical, Assignment, Increment and decrement, Bitwise and Special Operators

1.5 Decision Making and Branching : Decision Making with if statement, Simple if statement, The if else statement, The else if ladder, The switch statement, The ? : Operator

1.6 Decision making and looping : The for statement, The while statement, The do statement, Jumps in Loops, Labeled Loops





1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping :

- ✓ A loop is a way of repeating lines of code more than once. The block of code contained within the loop will be executed again and again until the condition required by the loop is dissatisfied.
- ✓ Types of Loop in Java:
 - ✓ for statement
 - ✓ while statement,
 - ✓ do – while statement,
 - ✓ jumps in Loops,
 - ✓ labeled Loops

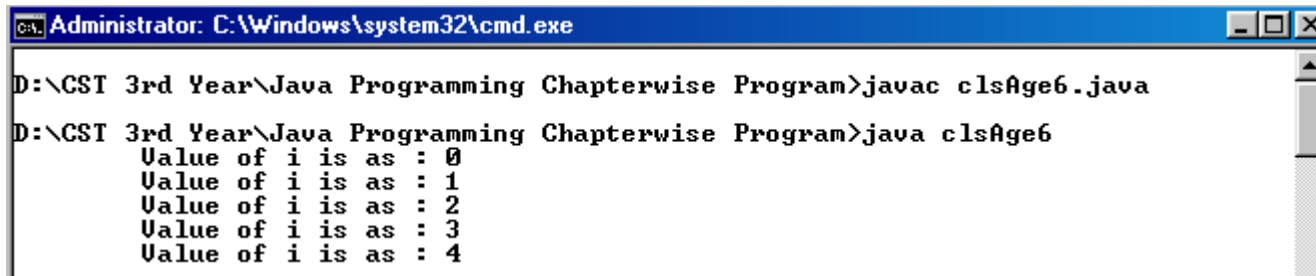




1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping using for Statement :

```
public class clsAge6{
    public static void main(String[] str){
        int i, n = 5;
        for( i = 0; i < n; i++){
            System.out.println("\tValue of i is as : " +
i);
        }
    }
}
```



```
Administrator: C:\Windows\system32\cmd.exe
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge6.java
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge6
    Value of i is as : 0
    Value of i is as : 1
    Value of i is as : 2
    Value of i is as : 3
    Value of i is as : 4
```

- **For Loop Explanation :** The for statement provides a compact way to iterate over a range of values. Programmers often refer to it as the “for loop” because of the way in which it repeatedly loops until a particular condition is satisfied.

- The general form of the for statement can be expressed as follows:

*for(initialization;; condition; increment) {
statement(s)
}*

- The *initialization* expression initializes the loop; it's executed once, as the loop begins.
- When the *condition* expression evaluates to false, the loop terminates.
- The *increment* expression is invoked after each iteration through the loop by incrementing *or* decrementing a value.





1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping :

- ✓ A loop is a way of repeating lines of code more than once. The block of code contained within the loop will be executed again and again until the condition required by the loop is dissatisfied.
- ✓ **Types of Loop in Java:**
 - ✓ for statement
 - ✓ while statement,
 - ✓ do – while statement,
 - ✓ jumps in Loops,
 - ✓ labeled Loops





1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping using while statement :

```
public class clsAge7{  
    public static void main(String[] str){  
        int x = 0;  
        while( x < 5 ){  
            System.out.println("value of x : " + x);  
            x++;  
        }  
    }  
}
```

```
Administrator: C:\Windows\system32\cmd.exe  
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge7.java  
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge7  
value of x : 0  
value of x : 1  
value of x : 2  
value of x : 3  
value of x : 4
```

• while Statement(s) :

The while statement continually executes a block of statements while a particular condition is true.

- Its syntax can be expressed as:

```
while (expression) {  
    statement(s)  
    increment / decrement;  
}
```

- It's an entry controlled loop, where the loop will execute if the condition satisfied.
- The while statement evaluates *expression*, which must return a boolean value.
- If the expression evaluates to true, the while statement executes the *statement(s)* in the while block.
- The while statement continues testing the expression and executing its block until the expression evaluates to false.





1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping :

- ✓ A loop is a way of repeating lines of code more than once. The block of code contained within the loop will be executed again and again until the condition required by the loop is dissatisfied.
- ✓ **Types of Loop in Java:**
 - ✓ for statement
 - ✓ while statement,
 - ✓ **do – while statement,**
 - ✓ jumps in Loops,
 - ✓ labeled Loops

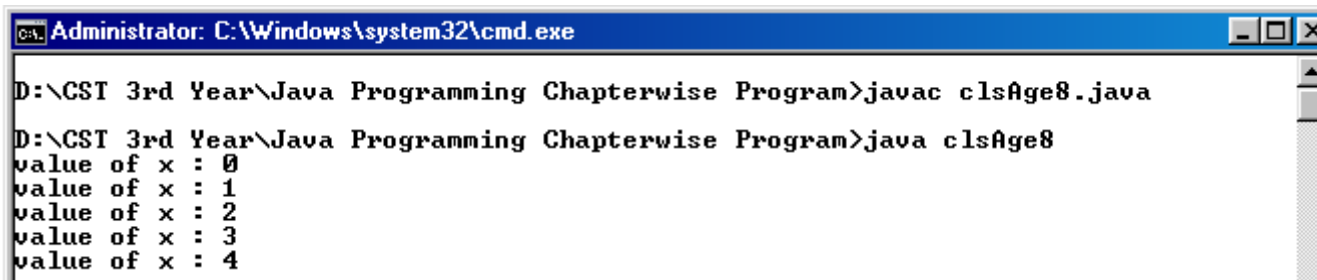




1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping using do- while statement :

```
public class clsAge8{  
    public static void main(String[] str){  
        int x = 0;  
        do{  
            System.out.println("value of x : " + x);  
            x++;  
        }while(x < 5);  
    }  
}
```



```
Administrator: C:\Windows\system32\cmd.exe  
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge8.java  
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge8  
value of x : 0  
value of x : 1  
value of x : 2  
value of x : 3  
value of x : 4
```

• do - while Statement(s) :

The do-while statement continually executes a block of statements while a particular condition is true.

- Its syntax can be expressed as:

```
do{  
    statement(s)  
    increment / decrement;  
}while( conditon );
```

- It's an exit controlled loop, where the loop will execute if the condition satisfied.
- The while statement evaluates *expression*, which must return a boolean value.
- If the expression evaluates to true, the while statement executes the *statement(s)* in the do-while block.
- The while statement continues testing the expression and executing its block until the expression evaluates to false.





1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping :

- ✓ A loop is a way of repeating lines of code more than once. The block of code contained within the loop will be executed again and again until the condition required by the loop is dissatisfied.
- ✓ **Types of Loop in Java:**
 - ✓ for statement
 - ✓ while statement,
 - ✓ do – while statement,
 - ✓ **jumps in Loops,**
 - ✓ labeled Loops

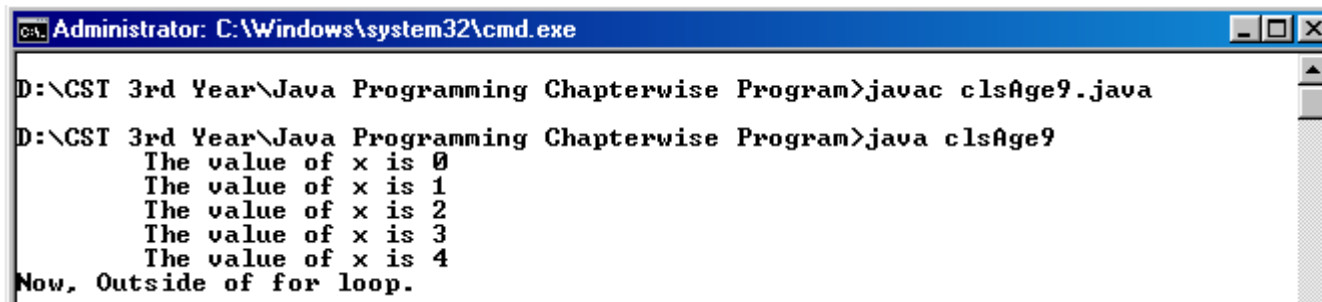




1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping jumps in loop(break):

```
public class clsAge9{
    public static void main(String[] args){
        for(int x=0; x<10; x++){
            if(x==5){ break;}
            System.out.println("\tThe value of x is " + x);
        }
        System.out.println("Now, Outside of for loop.");
    }
}
```



```
Administrator: C:\Windows\system32\cmd.exe
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge9.java
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge9
    The value of x is 0
    The value of x is 1
    The value of x is 2
    The value of x is 3
    The value of x is 4
Now, Outside of for loop.
```

• Jumps in Loop :

- The jumping statements are the control statements which transfer the program execution control to a specific statements.
- **Java has three types of jumping statements they are break, continue, and return.** These statements transfer execution control to another part of the program.
- **Java break statement can be used in the following cases :**
 - Inside the switch case to come out of the switch block.
 - Within the loops to break the loop execution based on some condition.
 - Inside labelled blocks to break that block execution based on some condition.

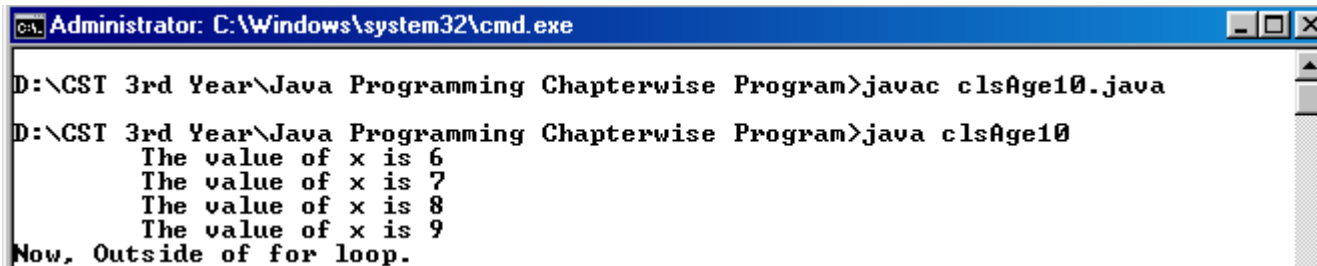




1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping jumps in loop(continue):

```
public class clsAge10{
    public static void main(String[] args){
        for(int x=0; x<10; x++){
            if(x <= 5){ continue; }
            System.out.println("\tThe value of x is " + x);
        }
        System.out.println("Now, Outside of for loop.");
    }
}
```



```
Administrator: C:\Windows\system32\cmd.exe
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge10.java
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge10
    The value of x is 6
    The value of x is 7
    The value of x is 8
    The value of x is 9
Now, Outside of for loop.
```

• Jumps in Loop :

- The jumping statements are the control statements which transfer the program execution control to a specific statements.
- **Java has three types of jumping statements they are break, continue, and return.** These statements transfer execution control to another part of the program.
- **Java continue statement can be used in the following cases :**
 - This is used within looping statements.
 - When the continue statement is encountered, then it skip the current iteration and the next iteration starts.
 - The remaining statements in the loop are skipped. The execution starts from the top of loop again.





1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping jumps in loop(return):

```
public class clsAge11 {  
    public static void main(String[] args){  
        for(int x=0; x<10; x++){  
            if(x<=5){  
                System.out.println("\tThe value of x is " + showInt(x));  
            }  
        }  
        System.out.println("Now, Outside of for loop.");  
    }  
    static int showInt(int i){  
        return i;    //  
    }  
}
```

```
Administrator: C:\Windows\system32\cmd.exe  
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge11.java  
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge11  
    The value of x is 0  
    The value of x is 1  
    The value of x is 2  
    The value of x is 3  
    The value of x is 4  
    The value of x is 5  
Now, Outside of for loop.
```

• Jumps in Loop :

- The jumping statements are the control statements which transfer the program execution control to a specific statements.
- Java has three types of jumping statements they are **break**, **continue**, and **return**. These statements transfer execution control to another part of the program.
- Java return statement can be used in the following cases :
 - A return statement causes the program control to transfer back to the caller of a method.
 - Every method in Java is declared with a return type.
 - A return type may be a primitive type like int, float, double, a reference type or void type(returns nothing).





1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping :

- ✓ A loop is a way of repeating lines of code more than once. The block of code contained within the loop will be executed again and again until the condition required by the loop is dissatisfied.
- ✓ **Types of Loop in Java:**
 - ✓ for statement
 - ✓ while statement,
 - ✓ do – while statement,
 - ✓ jumps in Loops,
 - ✓ **labeled Loops**





1.6 Introduction to Java : Decision Making & Looping

❑ Decision Making and Looping using Labeled Loop :

```
public class clsAge12{
    public static void main(String[] args){
        Outer:
        for(int i = 1; i <= 3; i++){
            inner:
            for(int j=1; j<=3; j++){
                if( i == 2 && j== 1){
                    break Outer;
                }
                System.out.println("\tValue of i & j is : "+ i + " : "+ j);
            }
        }
    }
}
```

```
C:\Administrator: C:\Windows\system32\cmd.exe
D:\CST 3rd Year\Java Programming Chapterwise Program>javac clsAge12.java
D:\CST 3rd Year\Java Programming Chapterwise Program>java clsAge12
    Value of i & j is : 1 : 1
    Value of i & j is : 1 : 2
    Value of i & j is : 1 : 3
```

• Labeled Loop:

- Here In the Java program, out for loop is named as “Outer” and inner for loop is named as “Inner”.
- If the condition for inner for loop satisfied then the it will come from the inner as well as outer loop and will also terminate the program.





Reference Book:

- Herbert Schildt - JAVA 2 : The Complete Reference
- Malhotra, Choudhary - Programming in Java
- Horstmann, Cornell - Core Java Vol I
- Liang - Introduction to Java Programming, 7e
- Deitel - Java for Programming
- Oracle Document - <https://docs.oracle.com/javase/tutorial/getStarted/index.html>

