

---

## UNIT 1 INTRODUCTION TO JAVA

---

Structure	Page No.
1.0 Introduction	
1.1 Objectives	
1.2 Introduction to Object Oriented Programming	
2.2.1 Objects and Class	
2.2.2 Data Hiding	
2.2.3 Abstraction	
2.2.4 Encapsulation	
2.2.5 Inheritance	
2.2.6 Polymorphism	
1.3 Features of Java Programming Language	
1.3.1 Use of Java in Industry	
1.4 Java Development Environment and Tools	
1.4.1 JDK	
1.4.2 JRE	
1.4.3 JIT and JVM	
1.5 Environment Setup for Java Application Development	
1.6 Summary	
1.7 Solutions/Answer to Check Your Progress	
1.8 References and Further Reading	

---

### 1.0 INTRODUCTION

---

It is a modern, high-level, evolutionary programming language that combines a well-designed structure with powerful features. In addition to the core language components, its software distribution is enriched with power supporting libraries for the purposes such as database, network, and GUI programming.

Java is a popular computer language with object-orientation concepts. To have a clear understanding of Java, we have to understand the reason behind its creation that forces and shapes the legacy that it inherits.

- Java is adaptive to the changing environments and uses. It supports the thinking of real-life situations in the form of objects and classes.
- To implement refinements and improvements in the art of programming in terms of data-centric applications development.

If we look at the history of the Java programming language, it was developed by James Gosling and his colleagues at Sun Microsystems in early 1990. Initially, it was started as a project called “Oak” with the goal to run it on a Virtual machine and have greater simplicity than C/C++. The first public implementation of Java 1.0 was released in 1995 with the goal of “Write Once, Run Anywhere”. Now, after many upgrades, we have Java 18 as the most updated version in the year 2022.

There are many application areas of Java programming language; including mobile application developments, it is considered the official language and compatible with well-known software Android Studio. Desktop GUI applications can be easily developed in Java programming language because of its Abstract Window Toolkit, Swing and JavaFX concepts. It is also suited for many other areas like Web-based

applications, Enterprise Applications, Scientific Applications, Gaming Applications etc. Its garbage collection module facilitates the 3-D game development.

This unit covered the foundation of object-oriented programming fundamentals and its founding concepts like Inheritance, Polymorphism, encapsulation, and abstraction. Further, we will discuss the Java programming language features that make it unique among all the programming languages. Also, robustness and portability are well-known properties of the Java programming language.

The Java development environment is an essential component for writing and executing programs. This includes various components like JDK, JRE, JVM, etc. Also, we will write and run our first Java program using Integrated Development Environment (IDE) in this unit.

---

## 1.1 OBJECTIVES

---

After going through this unit, you will be able to:

- Explain characteristics of Object-Oriented Programming,
- Explain features of Java,
- Explain the use of JDK, JIT and JVM, and
- Write and execute a simple Java program.

---

## 1.2 INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING

---

Java programming support concepts of classes and objects. Classes in Java can be considered as logical constructs upon which the entire java language is built.

Java programming languages have inherited object-orientation concepts from previously developed object-oriented programming languages such as C++ etc.. Some basic characteristics of Object Orientated programming, which you will extensively be using in this course of Java Programming, are explained here.

### 2.2.1 Objects and Class

An object is nothing but a real world entity that can be distinctly identified. Also objects poses their characteristics and features. A class is a template or blueprint that defines data fields and methods of objects of the same type.

Objects have the states and behaviours, while a class describes the behaviour/state of the object of its type support. When we look at real-world examples, cars, dogs, humans etc. are the objects. Dog shows its state and behaviour like name, breed and color are the state and barking, wagging the tail, and running are the dog's behaviour.

In figure1. 2, a circle class is shown with its three objects. We can see that each circle object is represented by the different radius. Here radius is the data field. Every object has its own data field value and methods implementation is common for all the objects. You will learn more about objects and classes in Unit 4 of this Block.

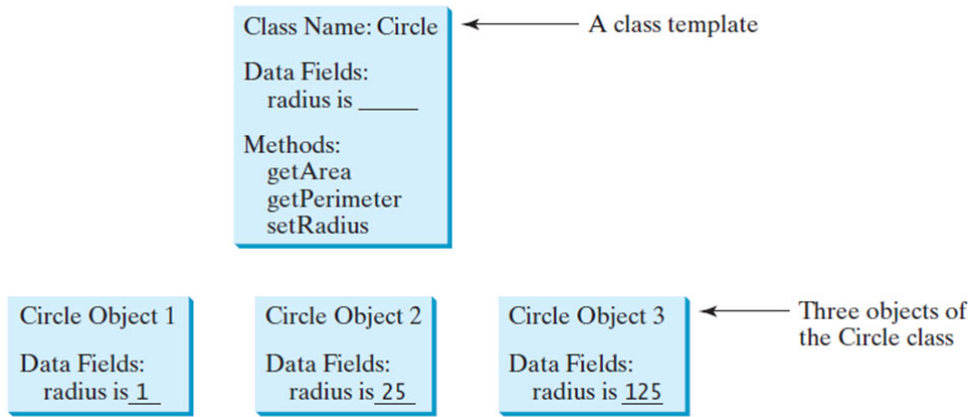


Figure 1.1: A class template along with objects

### 2.2.2 Data Hiding

Data hiding is one of the key aspects of Object-Oriented Programming that enables developers to protect their data and hide the implementation details from outside interference. To implement the data hiding, variables in a class should have a private access modifier. So when someone needs to access or modify the private variables from outside of the class, they should only be accessible by using the “get” and can be modified by using “set” methods.

### 2.2.3 Abstraction

Data abstraction is the process of hiding detailed information and only presenting the essential information to the user. For example, a car can be viewed as a car rather than its individual components. Data abstraction can also be seen as for identifying the important characteristics of an object by ignoring the useless details. In Java programming, abstraction can be achieved by using the abstract class/interfaces.

You can also apply the method abstraction to separate the use of the method from its implementation. Any user(client) can use a method without knowing about its implementation details. The details about the implementation are encapsulated in the method, and it will be hidden from the client who invokes this method. Further, if you change the implementation of a method, it will not affect the client program till you do not change the method signature.

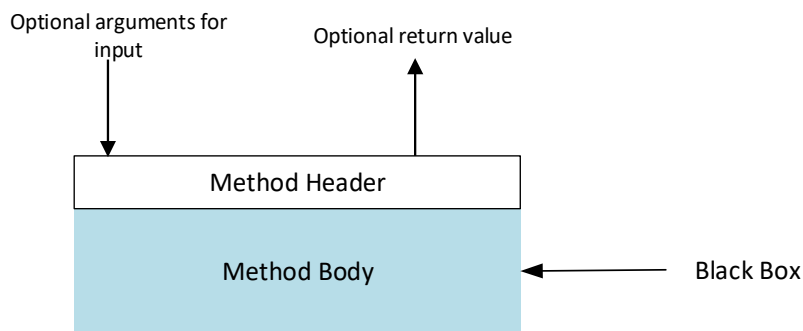


Figure 1.2: Method body as a Black Box

In figure 1.2, we can see that method body as a black box as its implementation is hidden from the user (client). For example when we use the *System.out.println* method

to display a string, we do not bother about how it is implemented. Throughout this course, we will see extensive use of the method `System.out.println`. As a user, we require to write the code to invoke the method that is the example method abstraction. The abstraction concept can be applied to the process of developing large programs; while writing the code, divide-and-conquer (also known as stepwise refinement) technique can be applied to decompose into subproblems. Similarly, class abstraction can separate the class implementation from how the class is used. The class owner describes the functions of the class and lets the user know how the class can be used.

### 2.2.3 Encapsulation

Encapsulation is another key concept used in object-oriented programming. It is a process of binding the data members and methods together. It is done in the form of a secure manner by making the data field accessible only to class members by declaring the data fields as private. The power of encapsulated code is that everyone outside of the class knows how to access and use the methods without thinking about implementation details. In Java, setter or getter methods are also used to ensure the read-only or write-only operations.

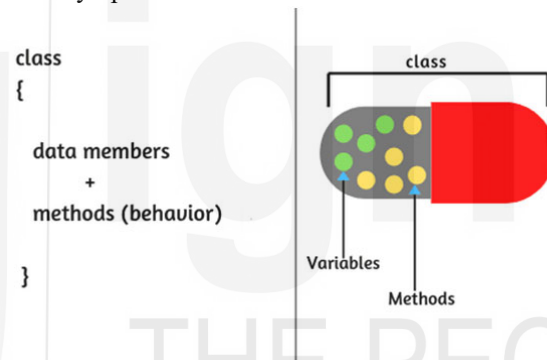


Figure 1.3: Encapsulation in Java

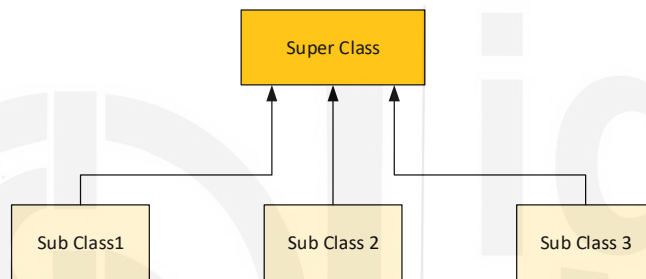
Figure 1.3 shows the encapsulation in Java, in which data members and methods are bound together and can be protected from outside classes.

Class abstraction and encapsulation are two sides of the same coin. Many real-life examples illustrate the class abstraction concept. Let's take an example of getting a loan, in which a specific loan can be seen as an object of the Loan class. The interest rate, loan amount, and loan period are its data fields. Calculating the monthly payment and total payments are its methods. The data fields are also known as data members. For example, when you take a car loan, a loan object is created by instantiating the class `Loan` with the loan interest rate, loan amount, and loan period. As a user of the `Loan` class, you are not required to know how these methods are implemented. You can use the methods to find your car loan's monthly payment and loan payment.

### 2.2.4 Inheritance

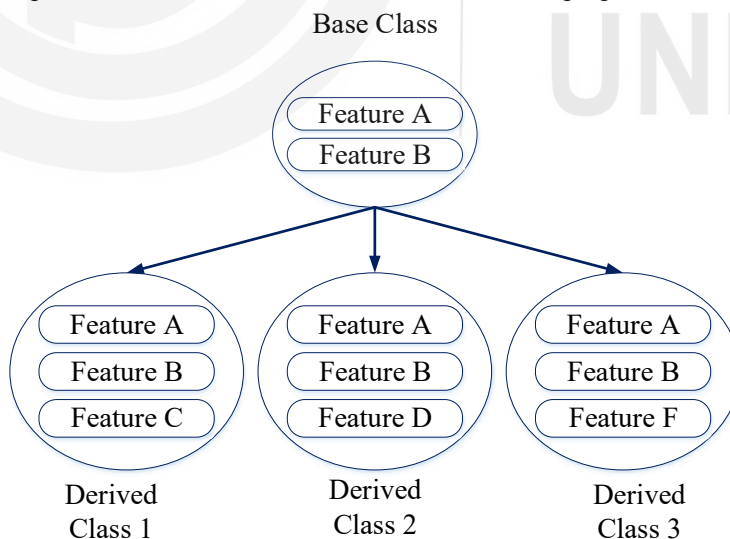
Inheritance provides reusability by inheriting the property in a new class from the derived class. It is a process by which one object acquires the property of another object. The Inheritance allows for the creation of a new class similar to the previously defined class, and the new class can have its own data members and methods. This feature of Java supports the concept of hierarchical classification.

Let us consider a Four-Wheeler class that contains data members and methods. The data members are engine capacity, number of gears, length, breadth, power steering, audio System, etc., and methods are forward movement, backward movement, stopping movement, etc. Let us assume that this Four-Wheeler class is defined and in use. Now a new class Car is needed to define which is similar to the defined class but needs some additional data members such as air conditioner, number of airbags, anti-breaking System and methods such as controlling air conditioner, automatic transmission, etc. The class Car need not include all data members and methods of the class Four-Wheeler. The class Car can reuse the features defined in class Four-Wheeler by inheriting it so that the Car class contains all data members and methods of the Four-Wheeler and data members and methods of its own. This new class Car alone needs to be defined and tested. This new class inherits all the data and methods from the base class ( in this case from class Four-Wheeler). The word Inheritance reflects how children inherit many of their characteristics from their parents. But the children also have their own characteristics. In OOP, a base class is seen as a Super Class( Parent class) and a derived class as a Sub Class (child class).



**Figure 1.4: Representation of Inheritance**

There are several advantages of Inheritance. It can reduce the development time of the program because reusability of the code requires less memory, less execution time, and redundancy of the code can be minimized. The real-life example of Inheritance is a child, and parents, like a child, can inherit all their father's properties.



**Figure 1.5: Inheritance Hierarchy**

Figure 1.5 shows the working of Inheritance; like you can see, three classes are derived from the base class. Features A, and B of the base class are inherited in all the three derived classes. Further, you can also see that derived classes have their own features like Feature C, Feature D, and Feature F in Derived Class1, 2, and 3,

respectively. Therefore, new classes use the concept of code reusability and extend their functionality. You will learn more about Inheritance and its implementation in Java in the next Block of this course.

### 2.2.5 Polymorphism

Polymorphism is the ability of an object to take many forms. It is a feature that allows one interface to be used for a general class of actions. In programming languages, polymorphism is the ability of an object to take on many forms. Polymorphism is derived from a Greek words: poly and morphs. The word “poly” means many and “morphs” means forms. So Polymorphism means many forms. Polymorphism is one of the important characteristics of OOPs. Many object-oriented programming languages support this characteristic, but their implementations vary from one object-oriented programming language to another object-oriented programming language. The polymorphism principle is implemented in java programming with method overloading and method overriding.

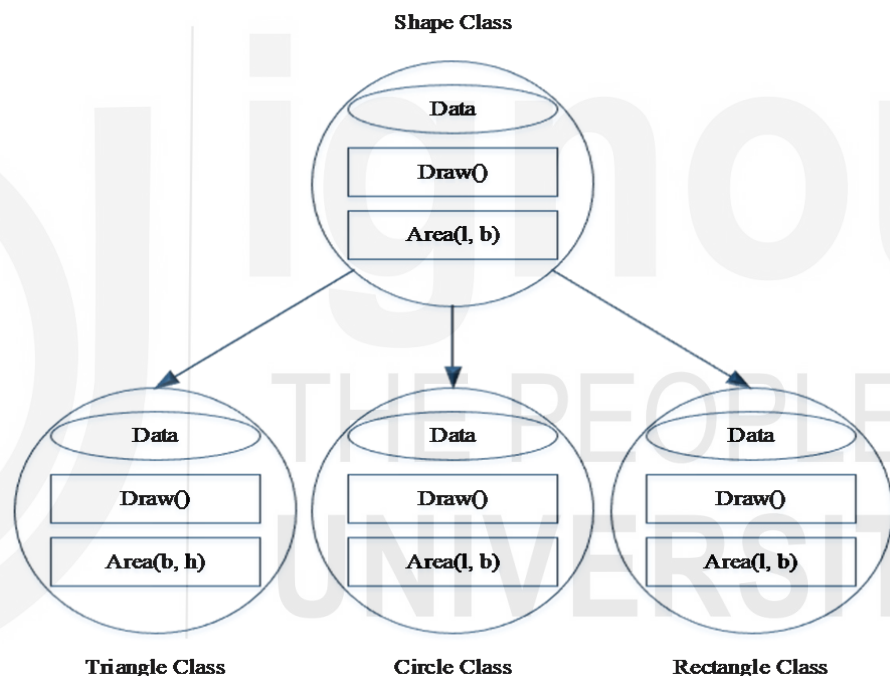


Figure 1.6 : Polymorphism

From figure 1.6, we can see that a single method draw () can be used to draw the different shapes like Triangle, Circle, and Rectangle. So here, we can observe that name of the function is the same, but the functionality( implementation) is different. This is known as function overriding, and it is a type of the polymorphism.

The polymorphism principle is divided into two sub-principles. They are:

- Static Polymorphism
- Dynamic Polymorphism

*Static binding* is also called as early binding or compiler time-binding of the method with object. Overloading is compile time Polymorphism where more than one method shares the same name with different parameters or signatures and different return types. The overloaded method should always be part of the same class, with the same

name but different parameters. The parameters may differ in their type or number, or both. Methods may have the same or different return types. Method overloading is the best example of static binding. In Method overloading, which method to be called is decided by the compiler at compile time.

*Dynamic binding* is also called *late binding* or *run-time binding*. Dynamic polymorphism is achieved at run-time. Here, Java compiler does not understand which method is called at time of compilation of the code. Only JVM decides which method is to be called at run-time. Method overloading and method overriding using instance methods are examples of dynamic polymorphism. Method overriding is the best example of dynamic binding. In method overriding, which method will be called decided at run-time. Dynamic method dispatch is a mechanism by which a call to an overridden method is resolved at run-time. This is how Java implements run-time polymorphism. When a reference calls an overridden method, Java determines the version of method to be executed based on the type of object it refers to. In the next Block of this course, you will learn the details of polymorphism and its implementation in Java.

### Check Your Progress – 1

1. What is a class? How is a class different from objects?

.....

.....

2. What is encapsulation?

.....

.....

3. What is Polymorphism? What are the different types of Polymorphism?

.....

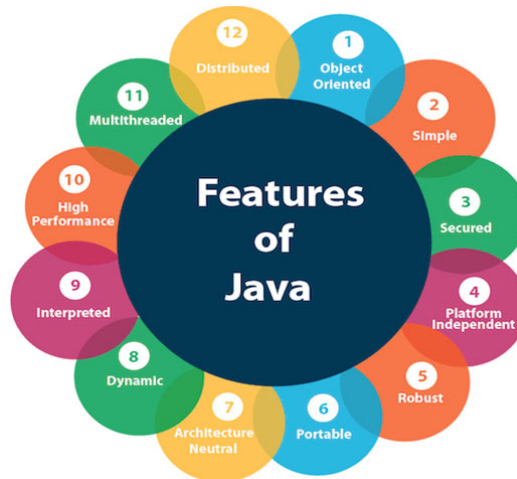
.....

---

## 1.3 FEATURES OF JAVA

---

Java is a well established popular programming language because of its popular features like portability and security. In addition, many other factors also played an essential role in moulding the final form of the language. Some important features of Java programming are briefly discussed in this section.



**Figure 1.7: Features of Java**

### **1) Object-Oriented**

Java is a programming language which is mainly based on object-oriented programming concepts, and we can say that everything in Java is an object. Object-oriented programming is so integral to Java that it is best to understand its basic principles before you begin writing even simple java programs. Objects have two fundamental purposes, which are as given below.

- To promote understanding of the real-world problems in a natural way.
- To provide practical basics for implementations of the design of the system.

An object contains data and methods which manipulate these data. For example, a student object may have the name, age, number, gender, course-name, university-name etc as data member, which represents the attribute of this object. The student object can perform a number of functions such as receiving this data, storing, finding the course-name, university-name etc. Also, in a system there can be many object students in a university with their unique identities.

People often hesitate to learn Java (OOP) because its learning curve is steeper than top-down programming (like C programming language). But once you learn the basics of OOP, you will find that it is easier and more intuitive approach for developing big projects or complex systems.

### **2) Simple**

Java was designed to be easy to understand and easy to write for the professional programmer. If you have some programming experience, you will be able to grab the java concept quickly. In case you have an understanding of object-oriented programming, then learning Java will be easy. The one who has C++, programming experience, learning Java will be pretty easy for them. Although Java is case sensitive but provides user-friendly syntax that can be understood easily.

### **3) Secure**

Java is one of the well-known programming languages for security. You can develop a virus-free system using java programming language because it has no explicit



pointer; it runs inside a virtual machine sandbox. Java also verifies the illegal Bytecode that can violate the access rights to the objects.

#### **4) Portable**

Portability is one of the significant challenges for internet applications because many computing devices and operating system are connected to the applications. Due to the different architecture of the devices/ operating system, running a single application on them is a challenge. Sometimes application code is required to be written considering platform-specific requirements when an application is developed using programming languages like C or C++. Java is an architecture-neutral programming language, and it is portable because it facilitates you to carry the java bytecode to any platform. With the help of bytecode, you can run the Java program on any computer virtually. More about bytecode you may learn from MCSL-210 course.

In general, Java provides three types of portability which are as follows.

##### **i. Source Code Portability**

When we run any given Java program, it must produce identical results regardless of the underlying CPU, operating system or Java compiler.

##### **ii. CPU Architecture Portability**

Currently, existing Java compilers produce object code (called Bytecode) to a CPU which physically does not yet exist. For executing the same Bytecode on different real CPU, a Java interpreter or Java Virtual Machine(JVM) is used. This nonexistent CPU ( i.e. JVM) allows the same object code to be executed in any CPU for which there is a JVM.

##### **iii. OS/GUI**

Similar to the JVM, which presents a virtual CPU, the Java libraries have a virtual operating system/GUI Java solves this problem by providing a set of library functions (in libraries provided by Java, such as awt, util and lang) that converse with an imaginary operating system and an imaginary GUI(Graphical User Interface). Each Java version provides libraries that implement this virtual operating system / GUI. The Java programs using these libraries easily provide the required port of operating system functionality and GUI.

#### **5) Robust**

Java provides an automatic garbage collector that runs on the virtual machine and destroys the objects which are not being used in the program anymore. The java designers paid enough attention so that the program could run reliably in various systems. While designing the Java, they focused on the program's robustness on the high priority. There are two main reasons to better understand the robust java programming behaviour: strong memory management and well-handled exceptions.

Exceptional conditions in traditional environments often arise in situations such as division by zero, file not found etc. they must be managed with clumsy and hard-to-read constructs. Java easily handles all these exceptions by providing object-oriented exception handling. In Block 3 of this course, you will learn more about Java Exceptions Handling.

## 6) Multithreaded

Java programming is designed to meet the real-world requirement of creating interactive, networked programs. For this, Java uses the concepts of multithreading to write the code, which can do so many things simultaneously. The Java run-time systems comes with an elegant solution for multiprocess synchronization that enables you to construct smoothly running interactive systems. You will learn more about the multithreading of Java in Block 3 of this course.

## 7) Architecture-neutral

Code longevity and portability were central issues for java designers. While designing Java, they have faced the issue that there is no guarantee that the program written today will run tomorrow—even on the same machine. Any update in the computer system can cause the malfunction of the system. The major goal was to “write once; run anywhere, anytime”. They designed Java Virtual Machine(JVM) with their hard efforts to overcome this situation. To a great extent, this goal was achieved.

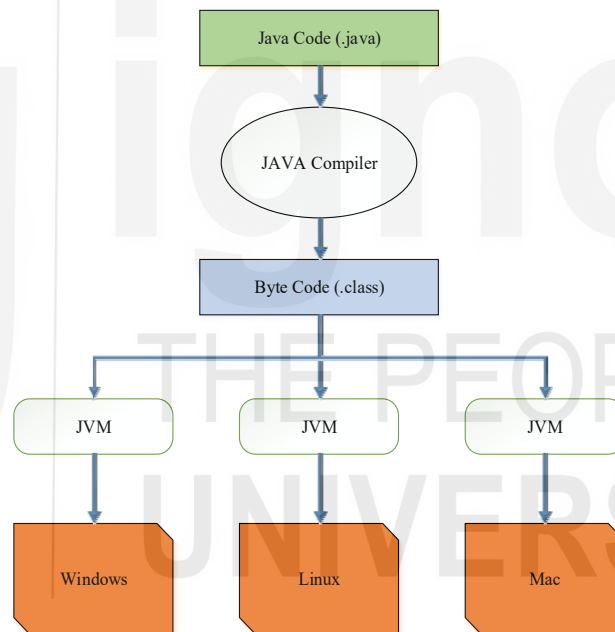


Figure 1.8: Java is Architecture-neutral

From figure 1.8, it can be seen clearly why Java programming is architecture-neutral. The only reason is its capability to generate the Bytecode, which runs on any JVM.

## 8) Interpreted and High performance

Java Bytecode can be executed on any machine that implements the Java Virtual Machine(JVM). It is designed in such a way that it can be translated into native machine code for every high-performance using a just-in-time compiler. Java run-time systems that present this feature lose none of the benefits of the platform-independent code.

You can achieve high performance because its *Just-in-Time compilers* are faster than the standard JVM. It also facilitates running one time and catching the results for

future uses. Also, it is possible to optimize the code over time. Therefore, we can say that it is a high-performance language.

## 9) Distributed

Java programming is designed for the distributed environment of the Internet because it handles TCP/IP protocols. Java programming also supports Remote Method Invocation (RMI), which enables a program to invoke methods across a network.

The reason behind saying Java is a distributed language is that after compiling the Java program on one machine, it can be transferred on another machine to execute because of its Bytecode facility. It will be beneficial for users who wish to use Remote Computers to execute their programs. Further, it can also facilitate the Local and Remote locations programmers to work together on the same project.

## 10) Dynamic

Java programs carry substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time. This enables to link code safely and expediently dynamically. It may affect the robustness of the java environment, in which a tiny fragment of Bytecode may be dynamically updated on a running system. Here dynamic just means changing something at run time that is not explicitly coded in the source code.

Another way in which Java is dynamic is the class loader, in which you can load a new already compiled class at run time in a standard language supported way. Java reflection method can also be used to operate on classes and instances specified by strings. For example, even without loading a new code, you can decide which already-loaded class you can use `Class.forName()` to get the class and instantiate an instance. If the string holding the name is read at the run-time, there's no way to know which class you'd be instantiating at compile time.

### 1.3.1 Use of Java in Industry

Java programming language has a significant impact on the industry, especially in the IT industry in India and globally. Many projects running for ICT support and services are developed using the Java programming language. In many applications, it is being used as HTTP request and HTTP response. A most famous “local server” with Java framework used is **tomcat**. It is the most popular Java application server. Another Java programming based tool for Big Data is HADOOP. It is an open-source tool built over Java programming and used to store and process a large volume of data. Java programming is used in many scientific applications, for example : MATLAB is one of the most famous scientific applications, its both front-end and back-end is written in Java. Many simulators including ClodSim are developed using Java technologies.

The Java programming language is being used to design and develop many web applications and mobile apps that may run on a single device or may be distributed among servers and clients in a network. Twitter is a well popular free social networking and micro-blogging Java-based service application being used widely. Many other applications like Operaminin, Signal, and Wikipedia Search are developed using Java programming language.

---

## 1.4 JAVA DEVELOPMENT ENVIRONMENT AND TOOLS

---

Java is a general-purpose computer programming language which is class-based, object-oriented and possess many other features. Its applications are typically, compiled to Bytecode and can execute on any java virtual machine (JVM). The latest version of Java is JDK 18, recently released in the year 2022.

Creating, compiling, and editing Java programs is flexible. You can extend it from one programming environment to another easily. In general, there are two basic approaches; a command-line environment, where the user types command and the computer responds, and another is an integrated development environment (IDE) in which we use the keyboard and mouse to interact with well designed graphical user interface( GUI) for writing, compiling and running the Java projects.

This section will discuss about the Java development environment and related considerations. More about IDE and execution of Java program you will learn in the course MCSL-210

### 1.4.1 JDK

JDK is an acronym for Java Development Kit. It is a software development environment which is used to develop Java applications. It physically exists and contains JRE along with development tools.

JDK is an implementation of any one of the below java platforms released by Oracle Corporation, which are as follows.

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

In general, JDK contains a private Java Virtual Machine (JVM) and a few other resources like interpreter/loader, java compiler, archivers (jar files) and Javadoc for the documentation generator to complete the development of Java Applications. Its software can be downloaded from the link: <https://www.oracle.com/java/technologies/downloads/> as given.

### 1.4.2 JRE

The Java Runtime Environment (JRE) is a software layer that runs on top of a computer's operating system software and provides the class libraries and other required resources to run a specific java program. JRE is one of the three interrelated components for developing Java programs along with JDK and JVM.

The JDK and JRE are the two components that interact with each other to create a sustainable run-time environment that enables the seamless execution of Java-based applications in virtually any operating system.

JRE run-time architecture generally uses the ClassLoader, Bytecode verifier, and Interpreter. In which ClassLoader dynamically load all the necessary classes to run a Java program. It also automates the loading of classes as per requirement, so it can also effectively utilize the memory. Bytecode verifier checks the format and accuracy of Java code before it passes to the interpreter. And finally, the interpreter helps to run the java program natively on the underlying machine.

### 1.4.3 JIT and JVM

JIT stands for *just in time compiler* and is an integral part of Java Virtual Machine(JVM) and Garbage Collector. It is capable of compiling the Java code into machine code, which can significantly improve the performance of Java Applications.

While the compilation of the codes, it is not guaranteed which code will be compiled. JIT usually compile the code as per its requirement to JVM, and once method execution crosses the certain limit (of approx. 10K execution), then it will be converted into machine code.

To run a simple program in Java programming, we have to use the Java compiler to convert the source code into Java class code. And then, after that, JVM (is responsible for running the program.

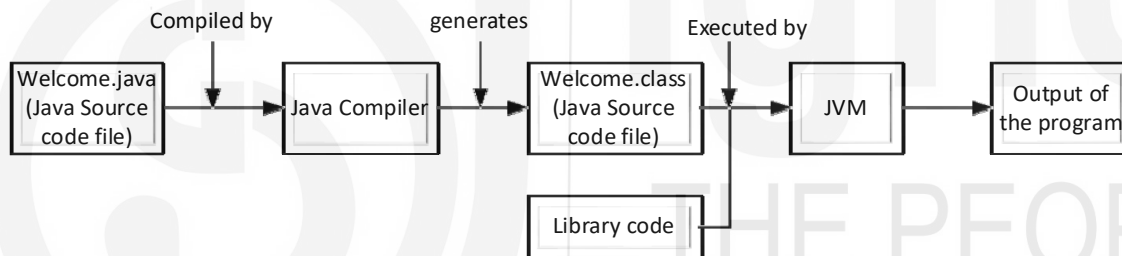
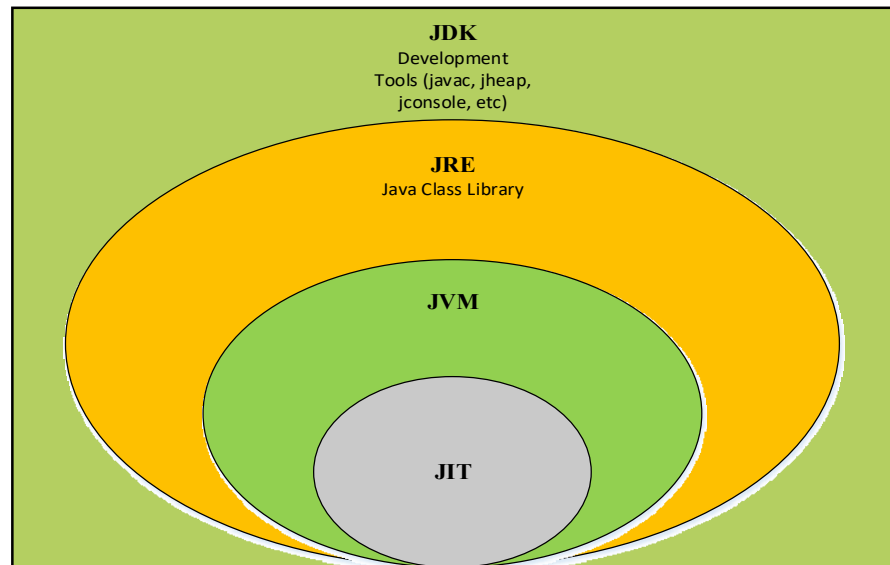


Figure 1.9: Flow of Java Program Execution

The role of JVM can be understood that a program written in a high-level programming language cannot run directly on any computer. First, it has to be translated into machine language using a compiler program. After being translated, it can run any number of times on one type of machine only. Another alternative way to compile the program is an interpreter, which translates the program instruction-by-instruction. It also has the capability to enable a machine language program to run from one computer to a completely different type of computer.

The designer of Java utilized the capability of both compilation and interpretation. Programs written in Java are compiled into machine language, but it is a machine language for a computer that does not really exist. This is what we call a “virtual” computer and is known as Java Virtual Machine(JVM). The machine language for the JVM is known as Java Bytecode. This code enables us to run the program on any computer.



**Figure 11.10: JIT, JVM, JRE and JDK in Java**

Java Development Kit (JDK) is enriched with many features; it has numerous libraries, tools, and frameworks. Once you run a Java program, it can be executed on any platform, which saves time. It also facilitates installing the Java platform on various Windows, Unix, and Mac operating systems.

In figure 11.10, we can see that every module of Java comes under JDK. JVM can be seen as an instance of JRE while executing the Java program, and it is widely known as a run-time interpreter. JVM is the cornerstone on top of which Java technology is built. It is a component which makes Java programming hardware & platform-independent. Just-in-time (JIT) compiler is the part of JVM that is used to speed up the execution time. It compiles the parts of the byte code simultaneously that have similar functionality.

---

## 1.5 ENVIRONMENT SETUP OF JAVA APPLICATION DEVELOPMENT

---

If you are writing your Java program in any editor like notepad etc. and going to run it with the help of command-line execution, you have to follow the step as given below.

**Step1:** Write a simple Java program; for example the Java code as given below.

```
class Welcome
{
    public static void main(String[] args)
    {
        //Display a welcome message on console.
        System.out.println("Welcome to Java.");
    }
}
```

Let's try to know the meaning of key terms/ words used in the above program: class, public, static, void, main, String [] is, System.out.println().

- `class` is a keyword which is used to declare a class in Java.
- `public` keyword is an access modifier(specifier) that decides the visibility of the method or data member. The `public` means it is visible to all.
- `static` is a keyword and if we declare any method as static, it belongs to the class. The core advantage of the static method is that there is no need to create an object to invoke the static method. The JVM executes the `main()` method, before creating any object, so it doesn't require to create an object to invoke the `main()` method, which also saves the memory.
- `void` is the return type of the method. It means that this method doesn't return any value.
- `main` represents the starting point of the program.
- `String[] args` or `String args[]` is used for passing the command-line argument.
- `System.out.println()` is used to print a statement. Here, *System* is a class, *out* is an object of the *PrintStream* class, *println()* is a method of the *PrintStream* class.

**Step2:** Now save the program using its class name as file name; for example, for the above program, it will be `Welcome.java`

**Step3:** Compile the program using the command given below.

→ `javac Welcome.java`

**Step4:** Run the program using the command by writing Java and then name of the class file. For example

→ `java Welcome`

Then it will print the output on the console like “Welcome to Java.”

Another way to write and run the java programs is to use the Integrated Development Environment (IDE), in which everything you need to create, compile, and run the program with the help of a single application. IDE also provides a graphical user interface.

NetBeans and ECLIPSE are two free and popular IDEs for developing Java programs. These IDEs are pretty simple and easy to understand. You are recommended to use one of them for developing the java programs.

### **ECLIPSE:**

The ECLIPSE IDE (integrated development environment) is one of the well-known java programming platforms. In the year 2022 itself, it has approximately one million downloads per month, making it one of the leading IDEs. ECLIPSE has many pre-packages and plug-ins for the support of java programmers.

### **NetBeans:**

NetBeans IDE is a free, open-source integrated development environment for application development on Windows, Mac, Linux, and Solaris operating systems. It provides intelligent overviews to help you manage, understand, and manage your

applications, including out-of-the-box support for popular technologies. NetBeans IDE offers tools for Java web, enterprise, desktop and mobile application development. This is the IDE which supports the latest version of JDK, Java EE and JavaFX.

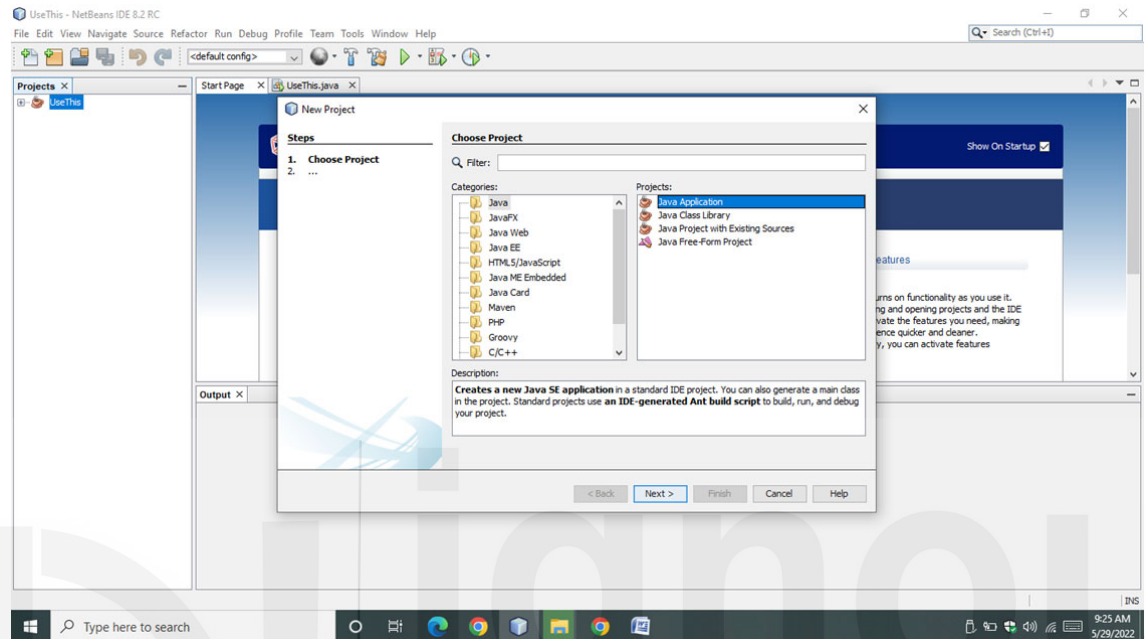


Figure 1.11: Netbeans IDE

A few key features of IDEs are as follows, which motivate programmers to use it.

- **Code completion or code insight**

The IDE is capable of providing the hint for keywords and function names which is quite crucial. It also helps us to remove the typographical errors and provide the suggestion with a list of appropriate functions based on the situation.

- **Resource management**

IDE is able to manage the resource effectively, especially the one required for library and header files. An IDE should be aware of any required resources so that errors can be spotted at the development stage, not at the time of compiling or building.

- **Debugging tools**

The program written using IDE is easy to debug; it may be able to give the variable values at certain points during the program's execution.

- **Compile and build**

In the programming language that requires a compile or build stage, IDEs translate the code from high-level language to the object code of the targeted platform. In general, IDE specializes in one language or a set of similar languages.

**Advantages of using IDE:** IDE can be advantageous in many aspects while writing and running a program. A few key advantages are as listed below.



### 1. Lest time and effort

The main purpose of an IDE is to make application development faster and easier. Its tools and features are developed to help programmers to organize resources, prevent mistakes, and provide shortcuts.

### 2. Enforce project or company standards

IDEs provide the same standard development environment for all its users. This provides a chance to a group of programmers to adhere to a standard way of doing things. Standards can be further enforced if the IDE offers predefined templates or if code libraries are shared between different team members/teams which is working on the same project.

### 3. Project management

Project management using IDE can be in twofold. First, many IDEs have documentation tools that either automate the entry of developer comments or may force developers to write comments in different areas. Second, simply by having a visual presentation of resources using GUI, it makes a lot easier to know how an application is laid out instead of traversing the file system.

In the following figure 1.12 you may see the screenshot of Netbeans IDE in which the program given above is written and executed.

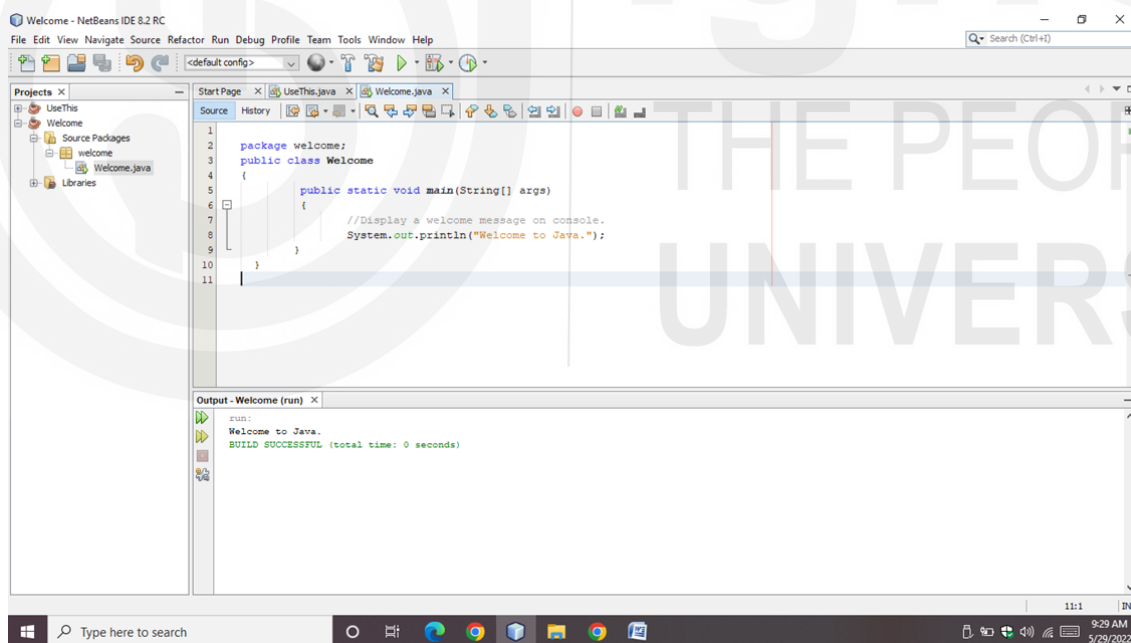


Figure 1.12: Writing and Running Java Program in Netbeans IDE

You will learn more about writing Java programs and running them in the MCSL-210 course. Here in this unit, you have a glimpse of how to write Java program. In the next unit, you will learn more about basics like variables/literals, data types, and operators and use them in writing Java programs.

## Check Your Progress – 2

1. What is JDK ?

.....

.....

2. What will be the output of the following code segment?

```
public class Example1
{
    public static void main (String args[])
    {
        System.out.println(" *****");
        System.out.println(" Let us Learn " +"Java Programming");
        System.out.println(" *****");
        System.out.println(" Use IDE to write Java Programamming" );
    }
}
```

.....

.....

3. What is the JIT compiler?

.....

.....

4. What are some available IDEs for developing Java Applications?

.....

.....

---

## 1.6 SUMMARY

---

Object-oriented programming takes a different approach to solving problems by using a program model that mirrors the real-world problems. It allows you to decompose a problem into subgroups of related parts easily. In this unit, we have discussed the basic features of object-oriented programming. Java is a very popular programming language because of its features. Some important features of Java, like platform-independent, portable, high performance, and distributed, making it popular among application developers. In this unit, we also discussed the Java development environment and tools.

---

## 1.7 SOLUTIONS/ANSWER TO CHECK YOUR PROGRESS

---

### Check Your Progress – 1

1. A class is a template or blueprint that defines data members and methods (member functions) of objects of the same type. On the other hand, an object is a real world entity that participates in problem-solving. Each object poses its attributes and features. Objects have the states and behaviours, while a class describes the behaviour/state of the object of its type support. When we look at real-world examples, cars, dogs, humans etc. are the objects. Dog shows its state and behaviour like name, breed and color are the state and barking, wagging the tail, and running are the dog's behaviour.
2. Encapsulation is a key concept of object-oriented programming. It is a process of binding the data members and methods together. The power of encapsulated code is that everyone outside of the class knows how to access and use the methods without thinking about implementation details.
3. Polymorphism is the ability of an object to take many forms. It is a feature that allows one interface to be used for a general class of actions. It means different actions can be performed using a single interface. In programming languages, Polymorphism is the ability of an object to take on many forms.

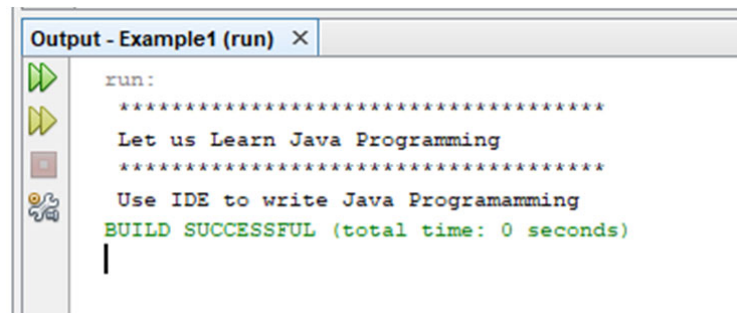
The Polymorphism is of two types:

- Static Polymorphism
- Dynamic Polymorphism

### Check Your Progress – 2

1. Java Development Kit (JDK) has the bundle of capabilities to run the Java programs smoothly and effectively. A few important points related to JDK are listed below.
  - JDK- JDK is the package that contains various tools, Compiler, Java Runtime Environment, etc. It is used for making java programs, we need some tools that are provided by JDK (Java Development Kit).
  - JRE - JRE provides the environment for the execution of the Java program. JRE contains a library of Java classes + JVM.
  - JVM (Java Virtual Machine) : It is a part of JRE that executes the Java program at the end. JVM converts Bytecode into machine-executable code to execute on a specific OS(hardware).

## 2. Output:



```
run:
*****
Let us Learn Java Programming
*****
Use IDE to write Java Programamming
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

3. JIT compilers can access dynamic run-time information, whereas a standard compiler does not and can make better optimizations like inlining functions that are used frequently. The JIT compiler compiles Bytecodes into native machine code at run time, which helps in improving the performance of Java programs.
4. An IDE for Java applications development is a software that allows Java developers to write and debug Java programs easily. It is a collection of various programming tools, accessible via a single interface along with several helpful features for developers, such as code completion and syntax highlighting. Some of the most popular Java IDEs are NetBeans, ECLIPSE, **IntelliJ** IDEA, and Codenvy.

---

## 1.8 REFERENCES AND FURTHER READING

---

- Herbert Schildt, “Java The Complete Reference”, McGraw-Hill, 2017.
- Savitch, Walter, “Java: An introduction to problem solving & programming”, Pearson Education Limited, 2019.