## ❖ Introduction to Java.

Java is a general purpose object oriented language developed by SUN Microsystems of USA in 1991. originally called **OAK**. James gosling, one of the inventors of the language, but was renamed "Java" in 1995.

Java was designed for development of software for consumer electronics devices like TVs, VCRs and Toaster and such other electronic machines.

The goal had a strong impact on the development team to make the language simple, portable and highly reliable.

The team members of java are James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sherdin.

All of the above developers thought that the existing languages like C & C++ had limitation in terms of both reliability and portability.

However, they modelled their new language Java on C and C++ but removed a number of features of C & C++ that were considered as sources of problems and thus made Java really simple, reliable, portable and powerful language.

## ❖ History of Java or Milestone of Java.

Following table lists some important milestones in the development of Java.

| YEAR | DEVELOPMENT STAGE |
|---|---|
| 1990 | Sun Microsystem decided to develop special software that could be used to manipulate consumer electronic devices. A team of Sun Microsystems Programmers headed by James Gosling was formed to undertake this task. |
| 1991 | After exploring the possibility of using the most popular object-oriented language **C++**, the team announced a new language named **"Oak".** |
| 1992 | The team, known as **Green Project team** by Sun, demonstrated the application of their new language to control a list of home appliances using a hand-held device with a tiny touch-sensitive screen. |
| 1993 | The **World Wide Web (WWW)** appeared on the Internet and transformed the text-based Internet into a graphical-rich environment. The **Green Project team** developed a new tiny program known as **applet** that could run on all types of computers connected to the Internet. |
| 1994 | The team developed a web browser called **"HotJava"** to locate and run applet programs on Internet. |

| | |
|---|---|
| 1995 | **Oak** was renamed **"Java"**, due to some legal snags. Java is just a name and is not an acronym. Many popular companies including Netscape and Microsoft announced to their support to Java. |
| 1996 | Java established itself not only as a leader for Internet Programming but also as a general-purpose, object-oriented programming language. SUN releases Java Development Kit 1.0 |
| 1997 | Sun releases Java Development Kit 1.1 (JDK 1.1) |
| 1998 | Sun releases the Java 2 with version 1.2 of Software Development Kit (SDK 1.2). |
| 1999 | Sun releases the Java 2 Platform, Standard Edition(J2SE) and Enterprise Edition(J2EE). |
| 2000 | J2SE with SDK 1.3 was released |
| 2002 | J2SE with SDK 1.4 was released |
| 2004 | J2SE with JDK 5.0 (instead of JDK 1.5) was released. This is known as J2SE 5.0. |

## ❖ Java Environment

Java environment includes a large number of development tools and hundreds of classes and methods. The development tools are part of the system known as **Java Development Kit** (JDK) and the classes and methods are part of the **Java Standard Library** (JSL), also known as **Application Programming Interface** (API).

1. **Java Development Kit:** The Java Development Kit **(JDK)** comes with a collection of tools that are used for developing and running Java Programs. They include:

| | | |
|---|---|---|
| appletviewer | -> | (for viewing Java applets) |
| javac | -> | (Java Compiler, which translates Java Sourcecode into bytecode files that the interpreter can understand.) |
| java | -> | (Java Interpreter, which runs applet and applications by reading interpreting bytecode files.) |
| javadoc | -> | (Creates HTML-format documentation from Java source code files.) |
| javah | -> | (Produces header files for use with native methods) |
| javap | -> | (Java disassembler, which enables us to convert bytecode files into a program description.) |
| jdb | -> | (Java debugger, which helps us to find errors in our programs.) |

## ❖ Define Term: Primitive Data Type

A data type which is directly operated on system is known as primitive data type. Such as int, float, double, long etc.

## ❖ Define Term: The ByteCode

**Bytecode** is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). It means the output of Java compiler is not executable code but it is bytecode.

## ❖ Why Byte code is necessary to execute the code of Java?

1. The reason is straightforward only the JVM needs to be implemented for each platform. Once the run-time package exists for a given system, any Java program can run on it. Remember, although the details of the JVM will differ from platform to platform, all interpret the same Java bytecode.

2. The execution of every Java program is under the control of the JVM, the JVM can contain the program and prevent it from generating side effects outside of the system.

3. The use of bytecode enables the Java run-time system to execute programs much faster than you might expect.

4. SUN supplies its **Just In Time(JIT)** compiler for bytecode, which is included in the Java 2 release. When the JIT compiler is part of the JVM, it compiles bytecode into executable code in real time, on a piece-by-piece, demand basis. It is important to understand that it is not possible to compile an entire Java program into executable code at all once, because Java performs various run-time checks that can be done only at run time. However the JIT compiles code as it is needed, during execution.

## ❖ What do you mean by JVM?
All language compilers translate source code into machine code for a specific computer. Java compiler also does the same thing. Java compiler done this by producing an intermediate code known as byte code for a machine that does not exist. This machine is called the **Java Virtual Machine** and it exists only inside the computer memory.

It is a virtual computer within the computer and does all major functions of a real computer. Figure shows the process of compiling a Java Program into bytecode which is also referred to as virtual machine code.

$$\begin{tikzcd}
\boxed{\text{Java Program}} \arrow[r] & \boxed{\text{Java Compiler}} \arrow[r] & \boxed{\text{Virtual Machine}}
\end{tikzcd}$$

**Process of Compilation**

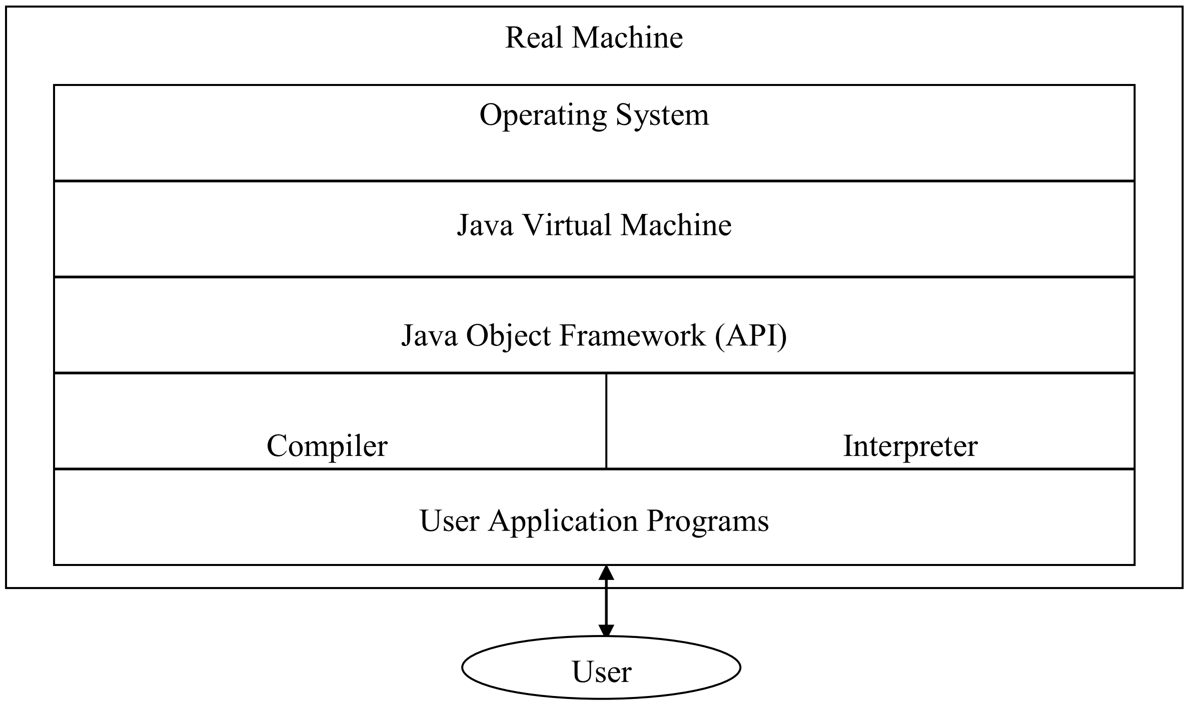
The virtual machine code is not machine specific. The machine specific code (known as machine code) is generated by the Java interpreter by acting as an intermediary between the virtual machine and the real machine as shown in the following figure. Remember that the interpreter is different for different machines.

$$\begin{tikzcd}
\boxed{\text{Bytecode}} \arrow[r] & \boxed{\text{Java Interpreter}} \arrow[r] & \boxed{\text{Machine Code}}
\end{tikzcd}$$

**Virtual Machine** **Real Machine**

**Process of converting bytecode into machine code**

Figure shows how Java works on a typical computer. The Java object framework (Java API) acts as the intermediary between the user programs and the virtual machine which in turn acts as the intermediary between the operating system and the Java object framework.



Layers of interaction for Java programs

## ❖ Java Buzzwords or Java features

The key consideration of Java describes by SUN Microsystems are as follow:

1 Simple
2 Secure
3 Portable
4 Object-oriented
5 Robust
6 Architecture-neutral
7 Multithread
8 Interpreted
9 High performance
10 Distributed
11 Dynamic

### (1) Simple

Java was designed simple way to develop program for professional programmer. Assuming that you have some programming experience, you will not find Java hard to master. If you already have knowledge of OOPS; it will be easy to learn Java. Because Java inherits the C/C++ syntax and many of the object oriented features of C++. Some of the more confusing concepts from c++ are eliminated in Java and implemented in a cleaner, more approachable manner.

### (2) Secure

Security becomes an important issue for a language that is used for programming on Internet. Threat of viruses and abuse of resources are everywhere. Java systems not only verify all memory access but also ensure that no viruses are communicated with an applet. The absence of pointers in Java ensures that programs cannot gain access to memory locations without proper authorization.

### (3) Portable

Java ensures portability in two ways.
1. Java compiler generates bytecode instructions that can be implemented on any machine.
2. The size of the primitive data types are machine independent.

### (4) Object-Oriented

Java is a true object-oriented language. Almost everything in Java is an object. All program code and data reside within objects and classes. Java comes with an extensive set of classes, arranged in packages that we can use in our programs by inheritance. The object model in Java is simple and easy to extend, while simple types, such as integers are kept as high-performance nonobjects.

### (5) Robust

Java is robust language. It provides many safeguards to ensure reliable code. It has strict compile time and run time checking for data types. So Java is strictly typed language.

To better understand how Java is robust, consider two of the main reasons for program failure:

1 Memory management mistakes
2 Mishandled exceptional conditions (that is, run-time errors).

1. **Memory Management:** It is very tedious task in traditional programming environments. For example, C/C++, the programmer must manually allocate and free dynamic memory. This sometimes leads to problems, because programmers will either forget to free memory that has been previously allocated or worse, try to free some memory that another part of their code is still using. Java virtually eliminates these problems by managing memory allocation and de allocation for you.

2. **Exceptional conditions:** Exceptional conditions in traditional environments often arise in situations such as division by zero or “file not found”. Java helps in this area by providing object-oriented exception handling. In well-written Java program, all run-time errors can and should be managed by your program.

### (6) Multithread

**Multithread** is a process by which more than two programs can be executed simultaneously. Java supports multithread programs. This means that we do not need to wait for the application to finish one tasks before beginning another.

**For example:**

(1) We can listen to an audio clip while scrolling a page and at the same time download an applet from distant computer. This feature improves the interaction performance of graphical applications.

(2) We can send tasks such as printing into the background and continue to perform some other task in the foreground. This approach would considerably improve the speed of our programs.

### (7) Architecture-Neutral

A central issue for the Java designer was that code longevity and portability.

One of the main problems facing programmers is that no guarantee exists that if you write a program today, it will run tomorrow-even on the same machine.

Operating system upgrades, processor upgrades and changes in core system resources can all combine to make a program fail. So, the Java designer's main goal was "**Write once; run anywhere, any time, forever.**" To great hard work this goal was achieved.

### (8) Interpreted

Normally, computer language is either compiled or interpreted. Java combines both these approaches thus making Java a two-stage system.

First Java compiler translates source code into special highly optimized instruction code known as **bytecode**. **Bytecode** are not machine instructions

Second stage Java interpreter generates machine code that can directly executed by the machine that is running on Java program. Therefore Java is both a compiled and an interpreted language.

### (9) High Performance

As described earlier, Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java bytecode. This code can be interpreted on any machine that provide Java Virtual Machine. Java, however was designed to perform well on very lower CPUS.

It is true that Java was engineered for interpretation, the Java bytecode was carefully designed so that it would be easy to translate directly into native machine code for very high performance by using a Just-In-Time compiler.

### (10) Distributed

Java is designed for the distributed environment of the Internet, because it handles TCP/IP protocols. You can also create an application on networks. It has the ability to share both data and programs. Java applications can open and access remote objects on Internet as easily as they can do in a local system. This enables multiple programmers at multiple remote locations to collaborate and work together on a single project. Java recharged these interfaces in a package called **Remote Method Invocation (RMI)**. This feature brings an unparalleled level of abstraction to **client/server programming**.

### (11) Dynamic

Java programs carry with them large amounts of run-time type information that is used to verify and determine accesses to objects at run time. This makes it possible to dynamically link code in a safe and convenient manner.
=================================================
=================================================

Q. **What is the meaning of each keyword in following statement in java?**
**public static void main(String args[])**

| | |
|---|---|
| Public | The keyword public is an access specifier that declares the main() as unprotected and therefore making it accessible to all other classes. So, main() must be declared as public, because it must be called by code outside of its class when the program is started. This is similar to the C++ public modifier. |
| Static | static declares this method as one that belongs to the entire class and not a part of any objects of the class. The main must always be declared as static because the interpreter uses this method before any objects are created. |
| Void | The type modifier void states that the main method does not return any value. |
| String args[] | Any information that you need to pass into main() is received by variables specified within the set of parentheses that follow the name of the method. These variables are called parameters. String args[] declares a parameter named args, which is an array of objects of the class String. Objects of type String store character strings. In this case, args receives any command-line argument present when the program is executed. |

**Lexical Issues:** Java programs are a collection of whitespaces, identifiers, comments, literals, operators, separators and keywords.

- **Whitespace:** Java is a free-form language. Free-form language means that you do not need to follow any special indentation rules. You can write java program in single line as long as there was at least one whitespace character between each token that was not already delineated by an operator or separator. In java **whitespace is a space, tab or newline**.

**For example**

```
class free_form
{
	public static void main(String arg[])
	{int i,len;len=arg.length;for(i=0;i<len;i++)
{System.out.println("Argument " +(i+1) +": " +arg[i]);}}}
```

- **Identifiers:** Identifiers are used for class names, method names and variable names. An identifier may be any descriptive sequence of uppercase and lowercase letters, numbers or the underscore and dollar-sign characters. It must not begin with a number. Java is case sensitive, so **NUM** is a different than **Num**.

Some examples of valid identifiers are:

| | | | |
|---|---|---|---|
| AvgTemp | count a5 | $test | this_is_ok |

**Invalid variable names include:**

| | | |
|---|---|---|
| 3count | high-temp | Not/Ok |

- **Literals:** A constant value in Java is created by using a literal representation of it. For example,

| | | |
|---|---|---|
| 1. | 100 | integer literal |
| 2. | 98.78 | floating literal |
| 3. | 'S' | character literal |
| 4. | "This is a test" | string literal |

- **Comments:** There are three types of comments defined by Java.

1. Single line comment **("//")**
2. Multiline comment **("*……*/")**
3. Documentation comment **("/**…..*/")**

**Purpose of documentation comment:** This type of comment is used to produce an HTML file that documents your program.

- **Separators:**

In Java there are a few characters that are used as separators. The most commonly used separator in Java is the semicolon. The separators are shown in the following table:

| Symbol | Name | Purpose |
|---|---|---|
| ( ) | Parentheses | 1. Used to contain list of parameters in method/function definition and when call it.<br>2. Used to defining precedence in expressions<br>3. Used as expression in control statements, loop statements and switch statements.<br>4. Used to surrounding cast types. |
| { } | Braces | 1. Used to initialize the value automatically in arrays. |
| [ ] | Brackets | 1. Used to declare array types and when dereferencing array values |
| ; | Semicolon | 1. Terminates statements |
| , | Comma | 1. Used to separates consecutive identifiers during variable declaration<br>2. Used to separates consecutive identifiers during variable declaration as arguments in function<br>3. Used to chain statements together inside a for loop. |
| . | Period | 1. Used to separate package names from subpackagees and classes.<br>2. Used to separate a variable or method from reference variable. |

- **The Java Class Libraries:** The Java environment relies on several built-in class libraries that contain many built in methods that provide support for such things as I/O, string handling, networking and graphics. Thus, Java as a totally is a combination of the Java language itself and its standard classes.

## ❖ Basic Concept of Object Oriented Programming(OOP):

It is necessary to understand some of the concepts used extensively in object oriented programming. These include:

1. Objects
2. Classes
3. Data Encapsulation and abstraction
4. Inheritance
5. Polymorphism

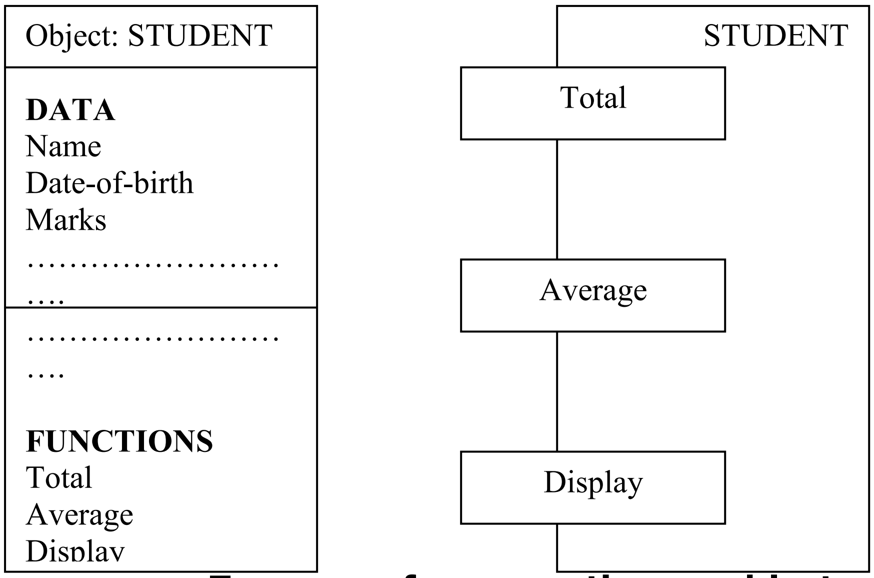
## ❖ Objects:

Objects are the basic run-time entities in an object-oriented system. It may represent a person, a place, a bank account, a table of data or any item that the program has to handle. They may also represent user-defined data such as vectors, time and lists. So, class variable is known as object.

Programming problem is analyzed in terms of objects and nature of communication between them. Objects take up space in memory and have an associated address like a record in Pascal or a structure of C.

When a program is executed, the objects interact by sending messages to one another. For example, if "customer" and "account" are two objects in a program, then the customer object may send a message to the account object requesting for the bank balance.

Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code. It is sufficient to know the type of message accepted, and the type of response returned by the objects. Figure shows two notations that are popularly used in object-oriented analysis and design.



**Two way of representing an object**

## ❖ Classes:

A class is a collection of objects of similar type. For example mango, apple and orange are members of the class fruit. Classes are user-defined data types and behave like the built in types of a programming language. The syntax used to create an object is no different than the syntax used to create an integer in C.

For example,

```
fruit mango;
```

will create an object mango belonging to the class fruit.

## ❖ Data abstraction and Encapsulation

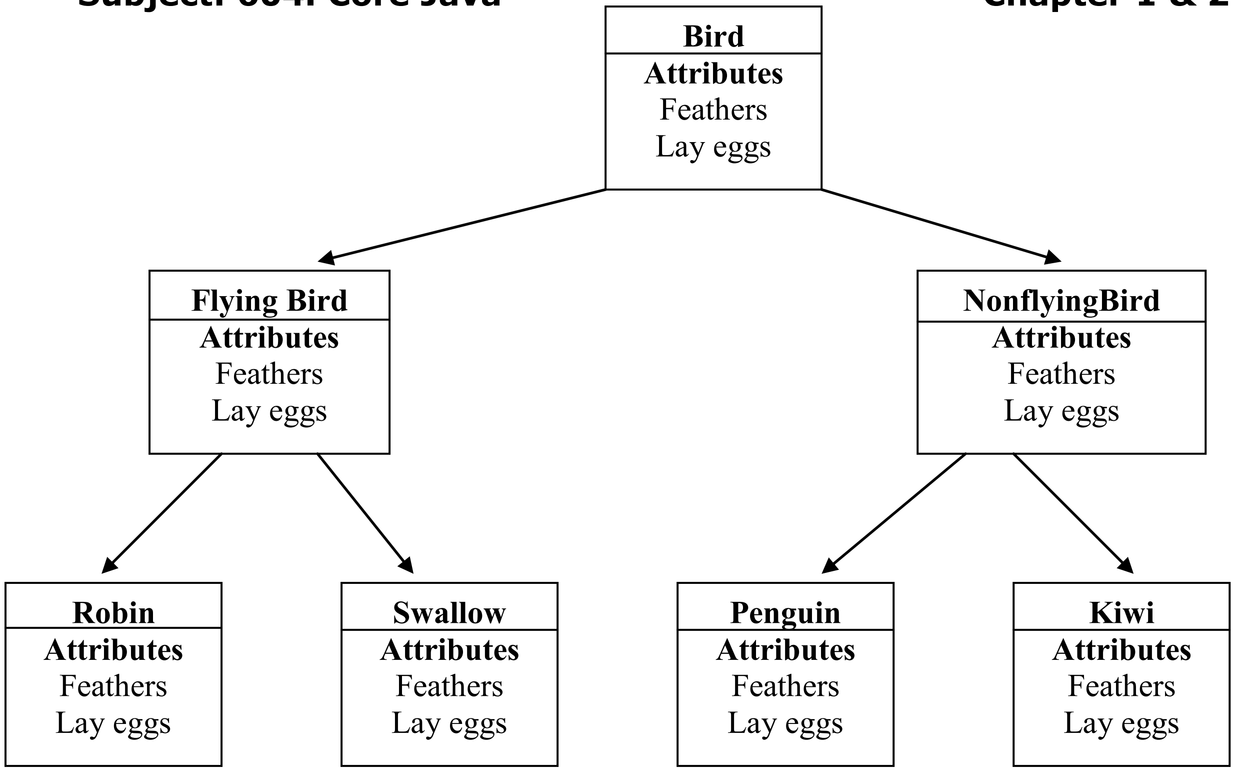
The wrapping up of data and functions into a single unit (called class) is known as **encapsulation**. Data encapsulation is the most striking features of a class. The data is not accessible to the outside world and only those functions which are wrapped in the class can access it. These functions provide the interface between the object's data and the program. This insulation of the data from direct access by the program is called **data hiding or information hiding**.

**Abstraction** refers to the act of representing essential features without including the background details or explanations. . **For example**, floating-point numbers are abstracted in programming language. You are not required to know how a floating point number is represented in binary while assigning a value to it.
Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and functions to operate on these attributes.

They encapsulate all the essential properties of the objects that are to be created. The attributes are sometimes called **data members** because they hold information. The functions that operate on these data are sometimes called **methods or member functions.**

## ❖ Inheritance:

Inheritance is the process by which objects of one class can share the properties of objects of another class. It supports the concept of hierarchical classification.



**Property inheritance**

In OOP, the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it. This is possible by deriving a new class from the existing one. The new class will have the combined features of both classes.

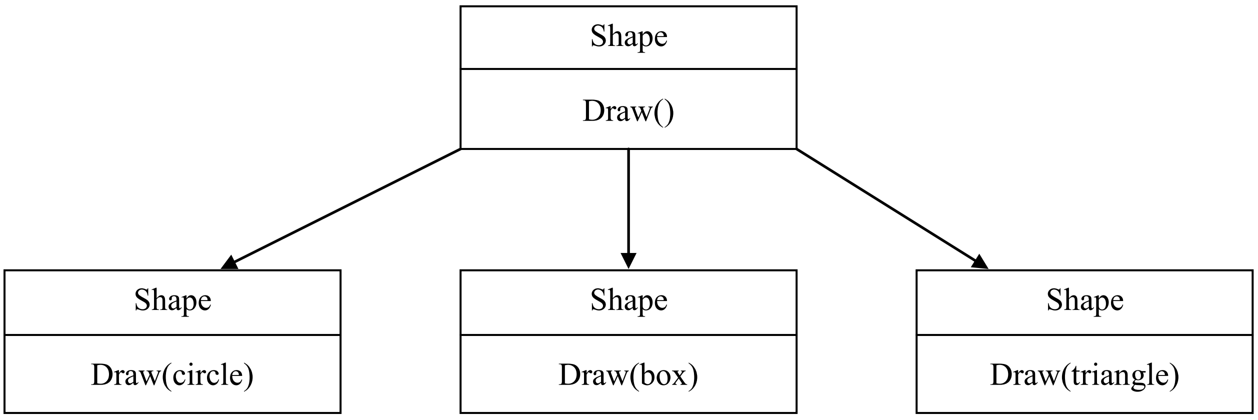
The real appeal and power of the inheritance mechanism is that it allows the programmer to reuses class that is almost but not exactly what he wants and to tailor the class in such a way that it does not introduce any undesirable size effects into the rest of class.

## ❖ Polymorphism:

Polymorphism means '**one name, multiple methods.**' The concept of polymorphism is implemented using the overloaded functions and operators.
**Operator Overloading** means an operator to exhibit different behaviors in different instances is known as operator overloading.

**Function overloading** means multiple functions with single name to perform different types of tasks are known as function overloading.

The following figure shows the idea of polymorphism



**Polymorphism**

## Q. Benefits of Java.

1. Free open source and there is not any development cost.
2. Garbage collection built into the language.
3. Write once Run anywhere (Byte code+ relative JVM)
4. No pointers
5. Well implemented Data Structure APIs (Collection)
6. Well Support APIs for web (JEE)
7. Well Support APIs for mobile (JME, partially Android)
8. No need to reinvent the wheel, lots of third party APIs available for support.
9. Support, and easy integration for many functional and scripting language like Python, Scala, Grovee etc.
10. Great Document.
11. Synchronization built into the language
12. Threading built into the language
13. Late linking makes some development and deployment smoother.
14. Platform-independent binaries, and rigorous code checking, make it possible to safely deploy code within another process (e.g. user-supplied builtins, application servers)