

Introduction of Java

History Of Java

- Java technology was created as a computer programming tool in a small, secret effort called "the Green Project" at Sun Microsystems in 1991 .
- A device-independent programming language code-named "Oak" was the result

History Of Java

1. Why Java came in market ?

- ‘write once, run everywhere’
- ‘hardware embedded’ + ‘internet compatible’ .

2. Business Objectives of Java

Java can work on databases, at front-end to implement security .

Applications

Today, we can find Java technology in

- ❑ Networks and devices that range from the Internet
 - Scientific supercomputers to laptops and
 - Cell phones
 - Wall Street market simulators
 - Home game players
- just about everywhere.

Java 2

- Java 2 Standard Edition
- Java 2 Micro Edition
- Java 2 Enterprise Edition

Java 2 SDK

- Java 2 Software Development Kit (Java 2 SDK)
 - Java Application or Applet Development
- Java 2 Runtime Environment (JRE)
 - Java Running Environment

Java Development Kit

Tools	Feature
javac	The Java Compiler
java	The Java Interpreter
Jdb	The Java Debugger
appletviewer	Tool to run the applets
javap	to print the Java byte codes
javaprof	Java profiler
javadoc	documentation generator
javah	creates C header files

.jar

- in software, **JAR** (**J**ava **A**Rchive) is an archive file format typically used to aggregate many Java class files and associated metadata and resources (text, images and so on) into one file to distribute application software or libraries on the Java platform.

- **Executing a JAR file**
- `Java -jar MyApp.jar`
- In order to create an executable JAR, one of the classes that we include in our JAR must be a main class.

Create a JAR file

```
jar cf MyApp1.jar C:\Java\MyApp
```

Extract content of JAR file

```
jar xf MyApp1.jar
```

Extract specific file from JAR file

```
jar xf MyApp1.jar Test1.class
```

JDK Versions

- JDK 1.02
- JDK 1.1
- Java 2 SDK v 1.2 (JDK 1.2)
- Java 2 SDK v 1.3 (JDK 1.3)
- Java 2 SDK V 1.4(JDK 1.4)
- Java 2 JDK v 5.0(instead of JDK 1.5)



...

IDE

(Integrated development environment)

IDE is a smart editor that equips a programmer with features like editing, compiling, deploying, debugging etc..

Java IDE Tools

- JBuilder
- Microsoft Visual J++
- Net Beans
- Eclipse
- Sun Java Workshop

- Etc.,

Java API

API includes hundred of classes and methods grouped into several packages

- Language support packages
- Utility packages (includes date ,time functions
- I/O Packages
- Networking packages
- AWT packages
- Applet Packages

Java Runtime Environment

JRE facilitates the execution of program in Java . It consists of

1. JVM – It Interprets java byte code to machine code and make it portable
2. Runtime class libraries –i.e. core class libraries that are requires for execution of java program
3. User Interface toolkit AWT and swing are example of toolkits

Features of java

1. Simple :- it can be learned easily
2. Object Oriented:- Any code in java is side a class
3. Distributed:- RMI packages is used for creating distributed server and client application.

Features of java

- 4. Interpreted:- Java is both compiled and interpreted with a compiler (javac).
- 5. Robust:- Java does not have permission to access computer memory.
- 6. Secure:- Java contains built-in security.

Features of java

7. Architecture Neutral (Platform-Independence):-
Java compiler produce the byte codes, byte codes are Platform – Independent.
8. Portable:- Java program can run on multiple platforms
9. High performance:- Just-in-Time (JIT) compiler-It can convert byte code into a machine code easily.

Features of java

- 10. Multithreaded:- It is the ability of an application to perform Multiple task at same time.
- 11. Dynamic:- Java supports dynamic memory allocation with automatic garbage collection.

Why Java Technology?

1. Write software on one platform and run it on practically any other platform
2. Create programs to run within a web browser and web services
3. Develop server-side applications for online forums, stores, polls, HTML forms processing etc.

Java better than C++ ?

- No Typedefs, Defines, or Preprocessor
- No Global Variables
- No Goto statements
- No Pointers
- No Unsafe Structures
- No Multiple Inheritance
- No Operator Overloading
- Destructors are replaced with finalize () function

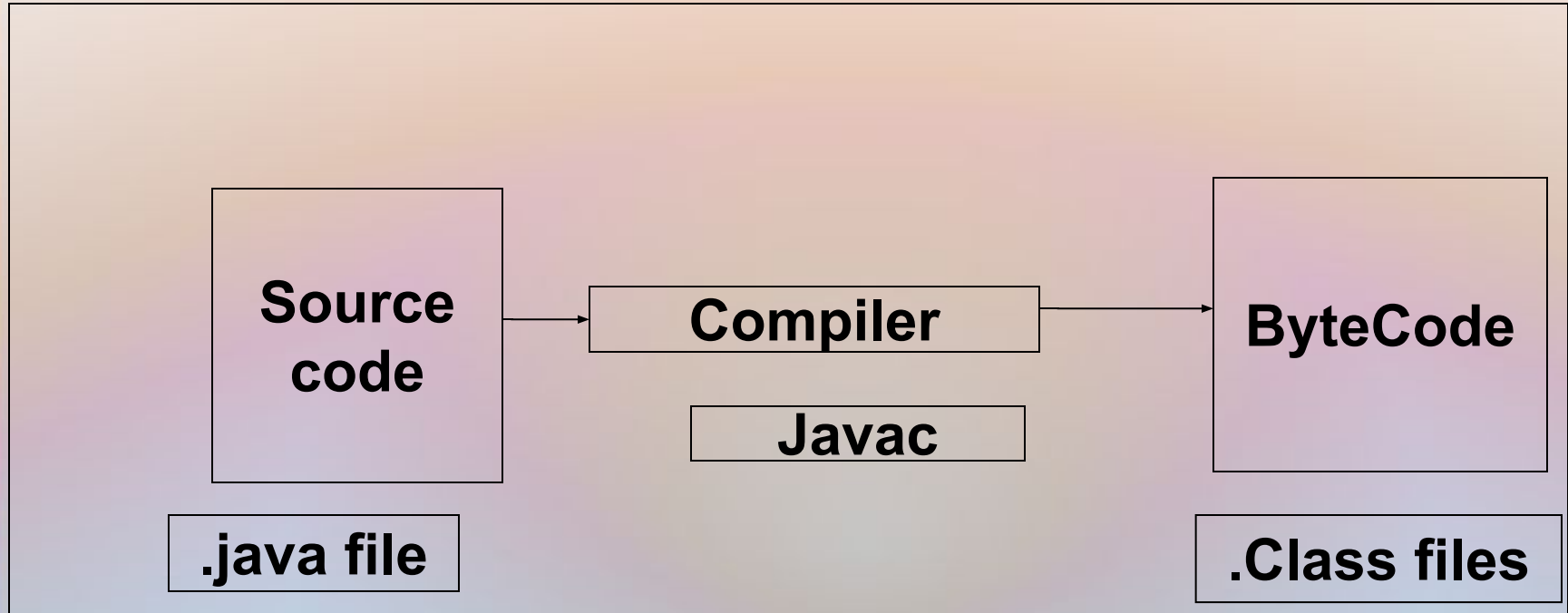
Added or Improved over C++

- Interfaces: type Vs. class
- Garbage collection
- Exceptions (More powerful than C++)
- String classes
- Package
- Multi-threads

Java Architecture

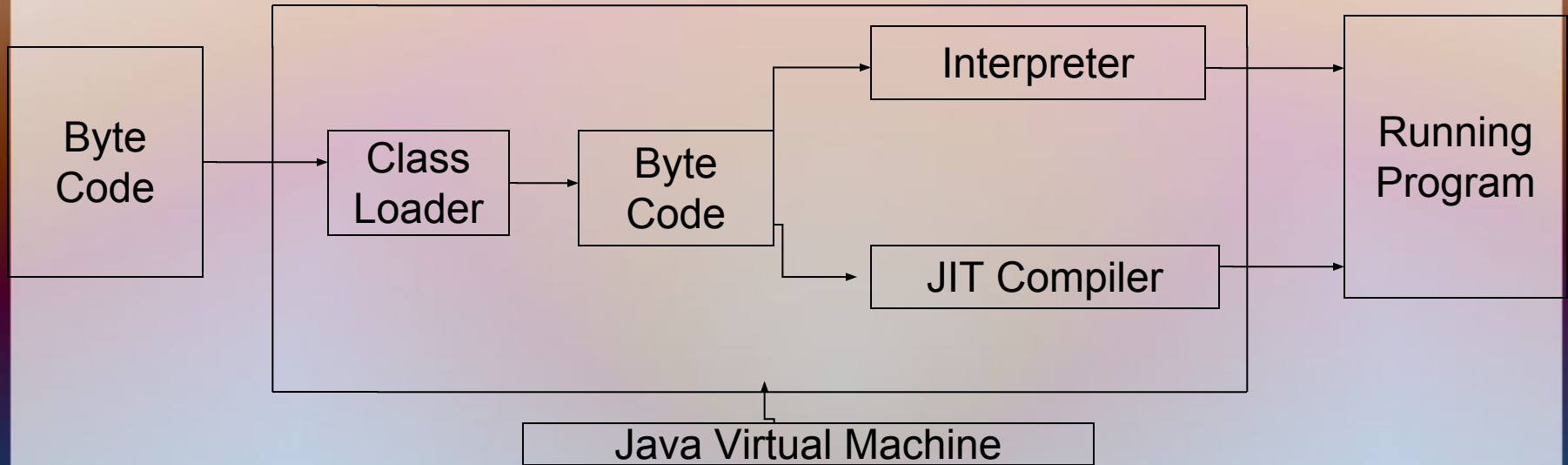
1. The Byte code
2. Java Virtual Machine
3. JIT (Just-in-Time compilers)

Process of compilation



Just-In-Time Compiler

Java Runtime System



- When java bytecode run, it is passed to run time environment interprets and checks the integrity and security of byte code then dynamically applies parameters , system configuration and environment variables .

Java virtual machine

JVM includes :

- Byte code Loader and java interpreter
- Byte code loader picks up .class file and pass it to java interpreter which translate the file into executable instructions for type of computer hardware and OS .

Java Applications

1. Application programs are mostly in the form of character user interface.
2. They are similar to other programs written in c or c++ .
3. An application is nothing but a program that runs on your computer, under the operating system of the computer
4. You can also create graphical user interface (GUI) applications.
5. These application are used in the windows

Object-Oriented programming features

Java is Object oriented programming language

.Features of OOPs are:

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

Classes

A *class* is a collection of objects of similar type. Once a class is defined any number of objects can be created which belong to that class.

How to create Class

Class name_of_the_class (User defined)

```
{  
//body  
}
```

Example:

```
class Television  
{  
//body
```


Object

An object is an instance of class. A software object maintains its states in variables and implements its behavior with methods declaring on object is similar to declaring a variable.

Syntax:

```
<class-name><object-name>= new class name();
```

E.g.,

```
Tv sony;          //declare
```

```
Sony=new Tv(); //create
```

```
Or Tv sony=new Tv(); //declare and create
```

How to create Object

Television tv; //declaration

tv= new Television();//memory allocation

Encapsulation

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

Encapsulation: public methods can be used to protect private data

Encapsulation

Storing data and functions in a single unit (class) is *encapsulation*. Data cannot be accessible to the outside world and only those functions which are stored in the class can access it.

Abstraction

Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes.

Inheritance

Inheritance is the process by which objects can acquire the properties of objects of other class. In OOPs *inheritance* provides reusability, like, adding additional features to an existing class without modifying it.

Polymorphism

Polymorphism means the ability to take more than one form. An operation may exhibit different behaviors in different instances. The behavior depends on the data types used in the operation. Polymorphism is extensively used in implementing Inheritance.

Some basics of the Java language

- Primitive types
 - byte, short, int, long; float, double, boolean, char
- Operators
 - arithmetic, logical, comparison, assignment
- Tests (i.e. conditions)
 - if (condition) then { ... } else { ... }
 - Operators: == != < <= > >= && || !
- Loops
 - for (initialization; test to continue loop; increment at end of loop) { ... }
 - while (test to continue loop) { ... }
 - do { ... } while (test to continue loop);

Keywords

abstract	continue	goto	package	switch
assert	default	if	private	this
boolean	do	implements	protected	throw
break	double	import	public	throws
byte	else	instanceof	return	transient

Keywords

case	extends	int	short	try
catch	final	interface	static	void
char	finally	long	strictfp	volatile
class	float	native	super	while
const	for	new	synchronized	

Java Standard Library (JSL)

1. Java.lang
2. Java.util
3. Java.io
4. Java.awt
5. Java.applet
6. Java.net
7. Javax.swing

Program to print hello message

```
//welcome1.java
```

```
public class welcome1
{
    public static void main(String args[])
    {
        System.out.println("Hello");
    }
}
```

Main method (in application)

- `public` : this is access specifier that declare main method unprotected and therefore making it accessible to all classes
- `static` : which declares method as one that belongs to entire class and not part of any object .
 - Main must be static because interpreter uses this method before any object is created.

Main method (in application)

- void : this states that main does not return any value .

Program 2

//printing a line with multiple statements

//welcome2.java

```
public class welcome2
{
    public static void main(String args[]){
        System.out.print("Welcome to");
        System.out.println("Java Programming!");
    }
```

Implementing Java program

- Installing java JDK
- Configuring java –setting environment variable
- Creating classname.java program
- Compiling (`javac classname.java`)
- Running program (`java classname`)

Data Types and Variables

Two data types available in Java:

- Primitive Data Types
- Reference/Object Data Types

Data types in java

1. Primitive data type

1. Numerical

1. Integer

2. Floating Point

2. Non Numerical

1. Character

2. Boolean

2. Non Primitive data type

1. Classes

2. Interface

- Integer
 - Byte
 - Short
 - Int
 - Long
- Floating
 - Float
 - Double

Table 1 Primitive Types

Type	Description	Size
int	The integer type, with range $-2,147,483,648 \dots 2,147,483,647$ (about 2 billion)	4 bytes
byte	The type describing a single byte, with range $-128 \dots 127$	1 byte
short	The short integer type, with range $-32768 \dots 32767$	2 bytes
long	The long integer type, with range $-9,223,372,036,854,775,808 \dots 9,223,372,036,854,775,807$	8 bytes
double	The double-precision floating-point type, with a range of about $\pm 10^{308}$ and about 15 significant decimal digits	8 bytes
float	The single-precision floating-point type, with a range of about $\pm 10^{38}$ and about 7 significant decimal digits	4 bytes
char	The character type, representing code units in the Unicode encoding scheme (see Advanced Topic 4.5)	2 bytes
boolean	The type with the two truth values <code>false</code> and <code>true</code> (see Chapter 6)	1 bit

Standard default values

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Number Literals

- `int i = 34;`
- `long l = 1000000;`
- `float f = 100.2f;`
or
`float f = 100.2F;`
- `double d = 100.2d`
or
`double d = 100.2D;`

1. Byte

- The byte data type is an 8-bit signed integer.
- Range -128 to 127 (inclusive).
- The byte data type can be useful for saving memory in large arrays,
- They can also be used in place of int where their limits help to clarify your code;

Reference Data Types:

- They are used to access objects.
- These variables are declared to be of a specific type that cannot be changed. For example, Employee, student
- Class objects, and various type of array variables come under reference data type.
- Default value of any reference variable is null.

Java Variables:

- Local Variables
- Class Variables (Static Variables)
- Instance Variables (Non static variables)

Program 3

```
//welcome3.java
```

```
//printing multiple lines with a single //statement
```

```
public class welcome3 {
```

```
    public static void main(String args[])
```

```
{
```

```
    System.out.println("welcome\n to\n Java\n Program  
!");
```

Program to perform addition of numbers

```
class addition
{
int a,b,c;
public void add()
{
a=10;
b=20;
c=a+b;
```

```
public static void main(String
args[])
{
addition a= new addition();
a.add();
}
}
```

```
System.out.println("sum="+c);
```

Control Statements

- Selection Statements
 - Using if and if...else
 - Nested if Statements
 - Using switch Statements
 - Conditional Operator
- Repetition Statements
 - Looping: while, do, and for
 - Nested loops

Selection Statements

- ☐ if Statements
- ☐ switch Statements
- ☐ Conditional Operators

if Statements

```
if (booleanExpression)
{
    statement(s) ;
}
```

Example:

```
if ((i >= 0) && (i <= 10))
{
```

```
    System.out.println("i is an " +
```

The if...else Statement

```
if (booleanExpression)
{
    statement(s) -for-the-true-case;
}
else
{
    statement(s) -for-the-false-case;
}
```


if...else Example

```
if (radius >= 0)
{
    area = radius*radius*PI;
    System.out.println("The area for the "
        + "circle of radius " + radius +
        " is " + area);
}
else
{
    System.out.println("Negative input");
}
```

Nested if Statements

Class marital

```
{  
public static void main(String args[])  
{  
System.out.print("Marital Status");  
int status=2;//married  
If(status==1)  
    System.out.print("Single");  
else if(status==2)  
    System.out.print("Married");  
else if(status==3)  
    System.out.print("Divorced");  
else
```

Conditional Operator

```
if (x > 0)
```

```
    y = 1
```

```
else
```

```
    y = -1;
```

is equivalent to

y = (x > 0) ? 1 : -1;

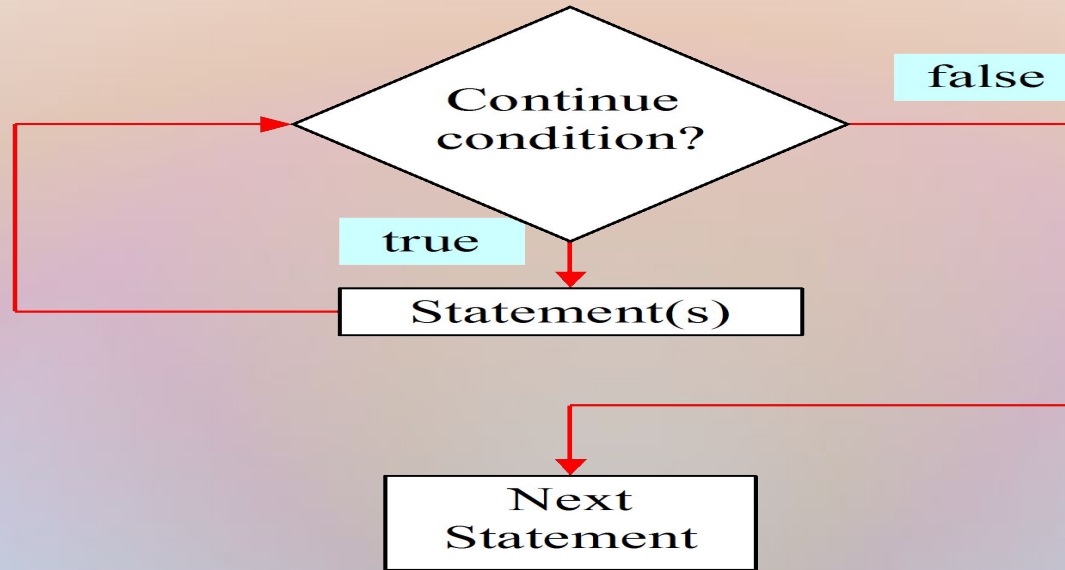
switch Statements

```
switch (year)
{
    case 7:    InterestRate = 7.25;
              break;
    case 15:   InterestRate = 8.50;
              break;
    case 30:   InterestRate = 9.0;
              break;
    default:
System.out.println("Wrong number of
years enter 7,15or 30");
```

Repetitions

- ☐ while Loops
- ☐ do Loops
- ☐ for Loops

while Loop Flow Chart



while Loops

```
while (continue-condition)
{
    // loop-body;
}
```

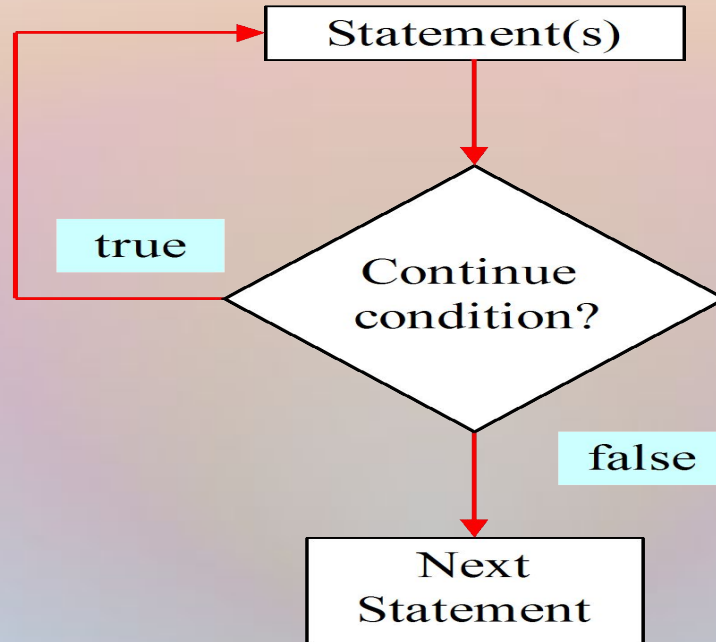
while Loop program

```
class whiletest
{
public static void main(String args[])
{
int n=1;
while(n<=10)
{
System.out.println(n+"");
n=n+1;
}
```


do Loops

```
do
{
    // Loop body;
} while (continue-condition);
```

do Loop



for Loops

Example:

```
int i;  
for (i = 0; i<100; i++)  
{  
    System.out.println("Welcome  
to Java! " + i);  
}
```

Enhanced for loop

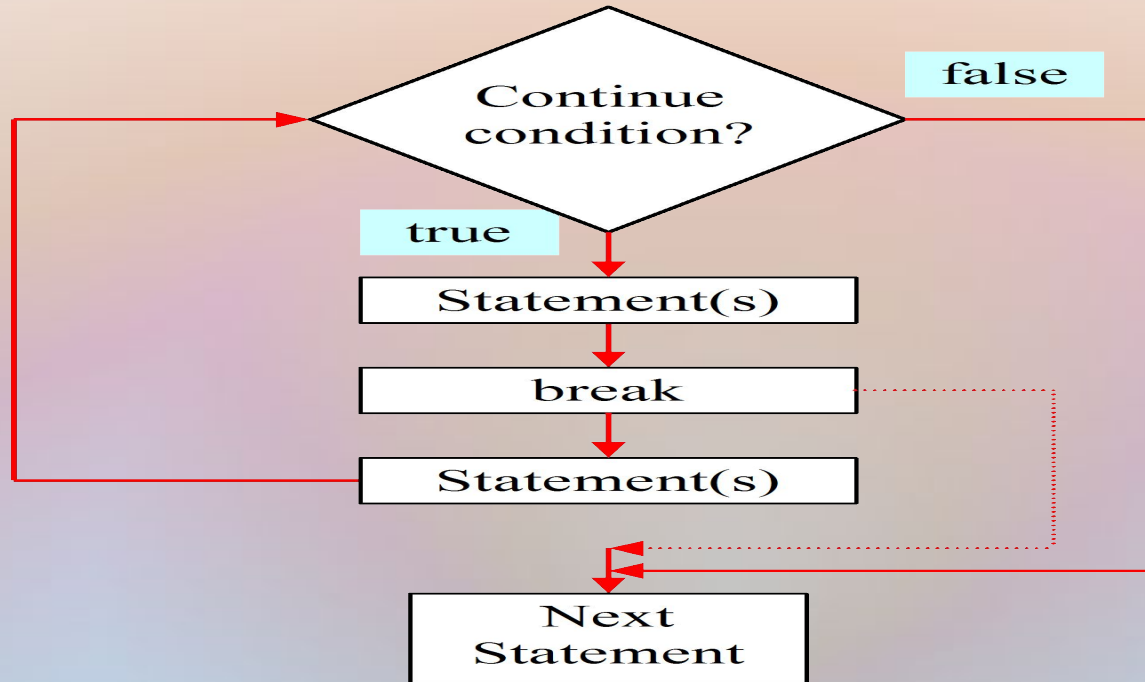
- For each loop
- It help us to retrieve the array elements efficiently rather than using array index .

Example

```
{  
    int r[]={23,45,56,67,78,5};  
    for( int k : r)
```

```
        System.out.println(k);
```

The break Keyword



Program on break

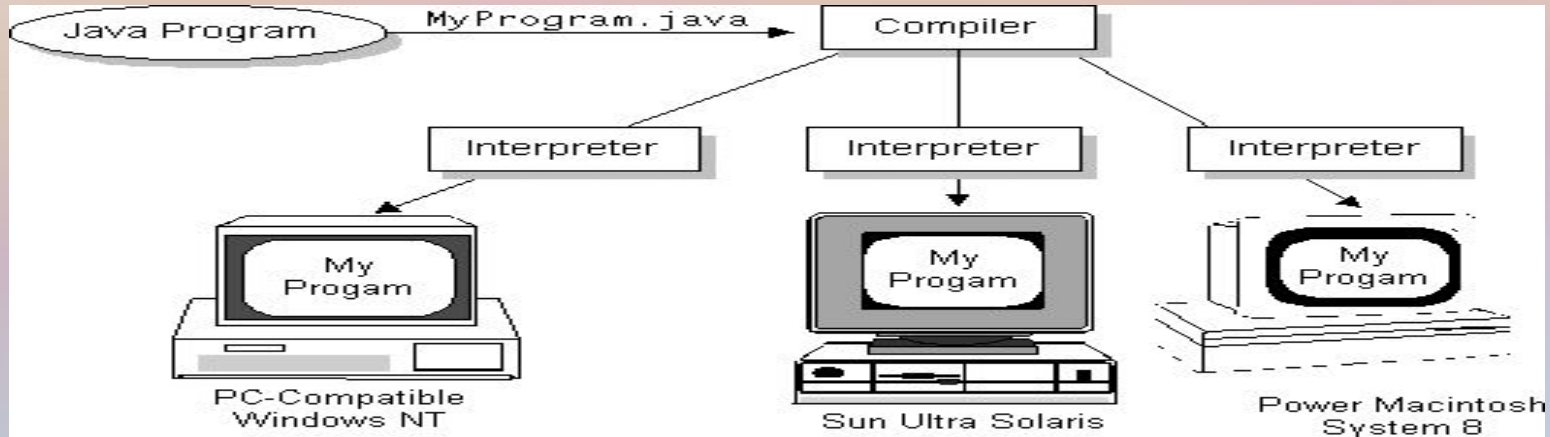
```
class testbr
{
public static void main(String args[])
{
for(int j=10;j<=100;j+5)
{
    if(j==50)
        break; //control to outer
System.out.println(j);
}
System.out.println("Break enabled");
```

Program on continue

```
class conti
{
    public static void main(String args[]) {
        for(int j=0;j<10;j++)
        {
            System.out.println("*"+j+" ");
            if(j%2==0)
                continue;
        }
    }
}
```

Notes:

- JVM will differ from platform to platform but still all understand same byte code(true)



note

- Byte -> smallest datatype in java
- Used specially when working with streams of data over file/ network
- Used for raw data
- When byte value is used in expression they are promoted to int

Note (int)

- In jdk 7 we can embed underscore in integer literal .
- Like `int y=123_456_33;`
- To make large value readable
- At time of compilation `_` will be discarded.
- Useful when encoding for things like phone/code
- `Int code= 011_22_23`

- How to check version of java running on your system?
- `java -version.`