RAMAIAH
Institute of Technology

Welcome to course on Programming in Java

Course Instructor

Hanumantharaju R

Assistant Professor

Dept. of CSE

hmrcs@msrit.edu

RAMAIAH
Institute of Technology

Course details

Prerequisites: C programming

Course Code: CSOE07-06

Course Name: Programming in Java

Course Credits: 3:0:0

Contact hours: 42

Programming language used: Java

RAMAIAH
Institute of Technology

Course Contents

Unit -1

Introduction to Java Programming: Java Buzzwords, Overview of Java Datatypes, Variables, arrays, Control statements. Java Programming Fundamentals: Object-Oriented Programming, the Three OOP Principles, Class Fundamentals, Declaring Objects, Assigning Object Reference Variables, Introducing Methods

Course Contents

Unit -2

Java Programming Fundamentals: Constructors, The this Keyword, Garbage Collection, The finalize( ) Method, Overloading Methods, Using Objects as Parameters, A Closer Look at Argument Passing, Returning Objects, Introducing Access Control, Understanding static, Introducing final, Introducing Nested and Inner Classes.

RAMAIAH
Institute of Technology

Course Contents

Unit -3

Inheritance, Packages & Interfaces

Inheritance Basics, Using super, Creating a Multilevel Hierarchy, When Constructors Are Called, Method Overriding, Dynamic Method Dispatch, Using Abstract Classes, Using final with Inheritance, Packages, Access Protection, Importing Packages, Interfaces, String and StringBuffer Handling.

RAMAIAH
Institute of Technology

Course Contents

Unit -4

Exception Handling: Exception-Handling Fundamentals, Exception Classes, Exception Types, Uncaught Exceptions, Using try and catch, Multiple catch clauses, Nested try Statements, throw, throws, finally.

Multithreaded Programming: Java Thread Classes, The Java Thread Model, The Main Thread, Creating a Thread, Creating Multiple Threads, Using isAlive() and join(), Thread Priorities, Synchronization, Suspending, Resuming and Stopping Threads.

RAMAIAH
Institute of Technology

Course Contents

Unit -5

Event Handling, Introducing Swing:
Two Event Handling Mechanisms, The Delegation Event Model, Event Classes, Sources of Events, Event Listener Interfaces, Using the Delegation Event Model, Adapter Classes, Inner Classes. Swing: Introducing Swing.

Lambda Expressions: Fundamentals, Block Lambda expressions, Passing Lambda Expressions as Argument, Lambda Expressions and Exceptions, Method References.

Textbooks

1. The Complete Reference - Java, Herbert Schildt 10th Edition, 2017, TMH Publications, ISBN: 9789387432291.

RAMAIAH
Institute of Technology

Reference book

1. Head First Java, Kathy Sierra and Bert Bates, 2nd Edition, 2014, Oreilly Publication , ISBN : 978817366602

Course Outcomes

This course uses assigned readings, lectures, and homework to enable the students to:

1. Examine the fundamental elements of object model and identify classes and objects for object-oriented programming. (PO-2,3,5 PSO-3)
2. Explore the OOP principles and basic constructs of Java language. (PO-2,3,5 PSO-3)
3. Develop java programs using inheritance, interfaces, and packages. (PO-2,3,5 PSO-3)
4. Explore the exception handling mechanism and thread synchronization. (PO-2,3,5 PSO-3)
5. Design the GUI application using swings and handle the interactions. (PO-2,3,5 PSO-3)

RAMAIAH
Institute of Technology

Scheme of Evaluation

CIE-30%- 2 tests of 30 marks each

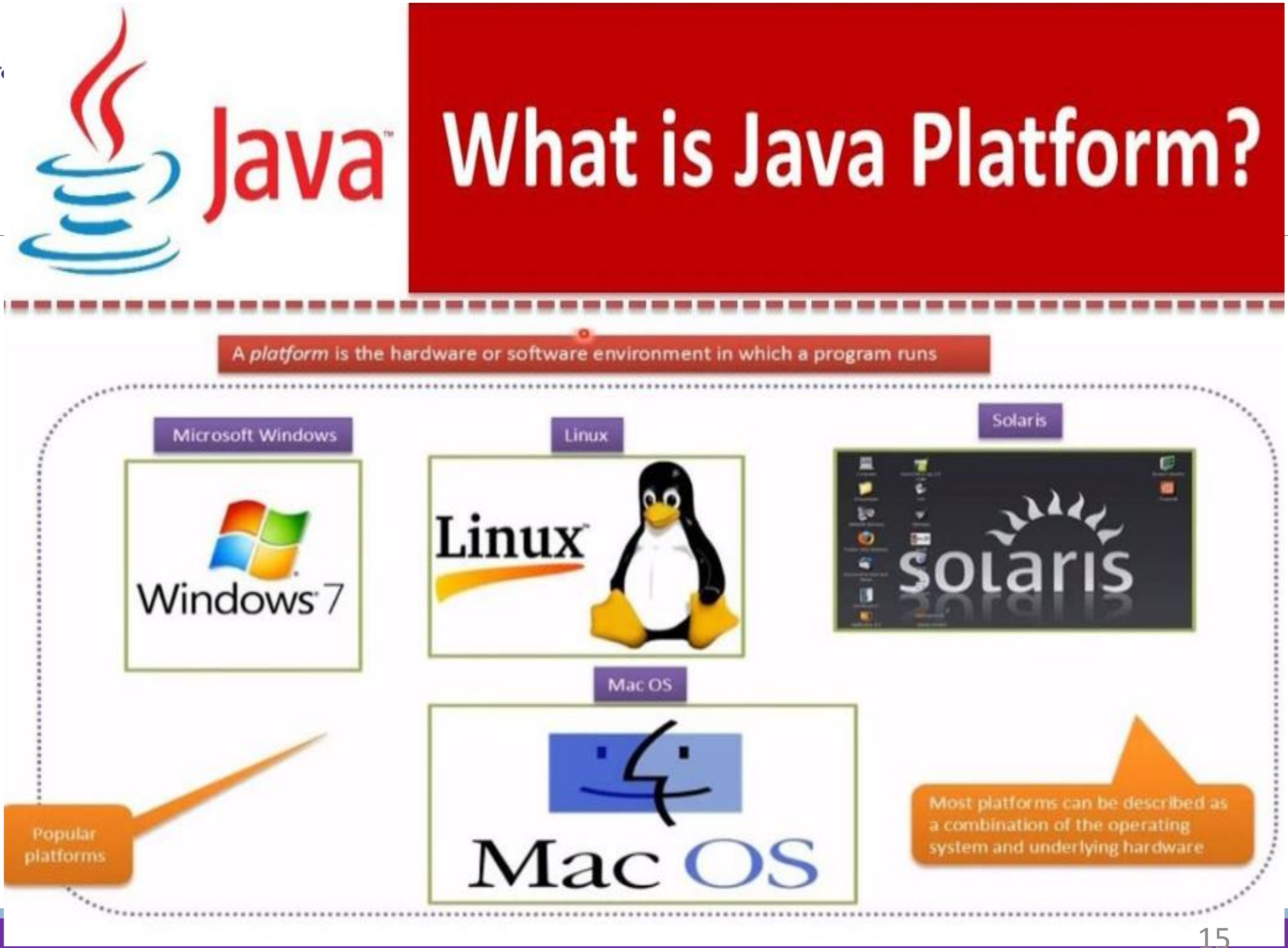
Non CIE-20%- 2 Quizzes of 10 mark each

SEE-50% - Exam for 100 marks

Introduction to Java

Java an Introduction

Java is a powerful programming language built to be secure, cross-platform and international, but Java is being continuously extended to provide language features and libraries that elegantly handle problems that are difficult in traditional programming languages, such as multithreading, database access, network programming and distributed computing. Java allows client-side programming via the applet.



What is Java Platform?

A *platform* is the hardware or software environment in which a program runs

Microsoft Windows

Linux

Solaris

Mac OS

Popular platforms

Most platforms can be described as a combination of the operating system and underlying hardware

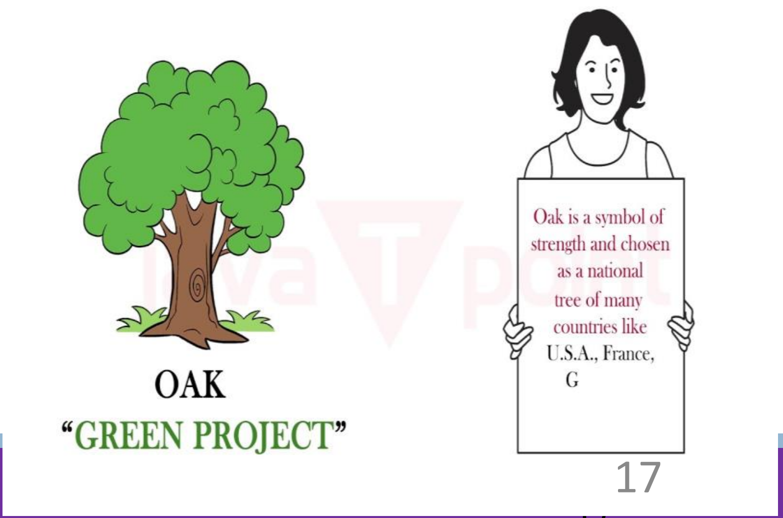
What is Java

- Java is an object-oriented programming language developed by Sun Microsystems, a company best known for its high-end Unix workstations. Modeled after C++, the Java language was designed to be small, simple, and portable across platforms and operating systems.
- Java is also a **platform**.
  - Any hardware or software environment in which a program runs, is known as a platform. Since Java has its own runtime environment (JRE) and API, it is called platform.

RAMAIAH Institute of Technology

History

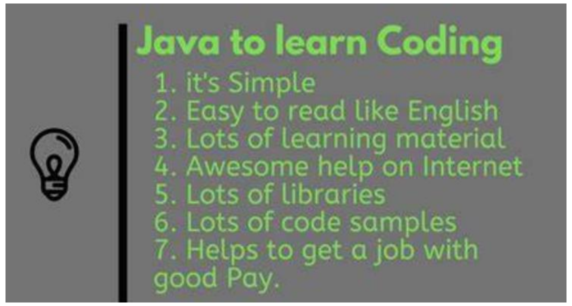
- **James Gosling**, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project
  in June 1991 at Sun Microsystems. The small
  team of sun engineers called **Green Team.**
- After that, it was called **Oak** and was developed as a part of the Green project.
- **Oak** was renamed as JAVA on May 20, 1995, by Microsystems.
- HotJava
  - The first Java-enabled Web browser
- JDK Evolutions
- J2SE, J2ME, and J2EE


OAK
"GREEN PROJECT"
Oak is a symbol of strength and chosen as a national tree of many countries like U.S.A., France, G

RAMAIAH Institute of Technology

Why to Learn Java?

- Java Is Platform-Independent
  - Platform-independence is a program’s capability of moving easily from one computer system to another
- Java Is Object-Oriented
  - Java is modeled after C and C++
- Java Is Easy to Learn
- There are no pointers in Java, nor is there pointer arithmetic. Strings and arrays are real objects in Java. Memory management is automatic.



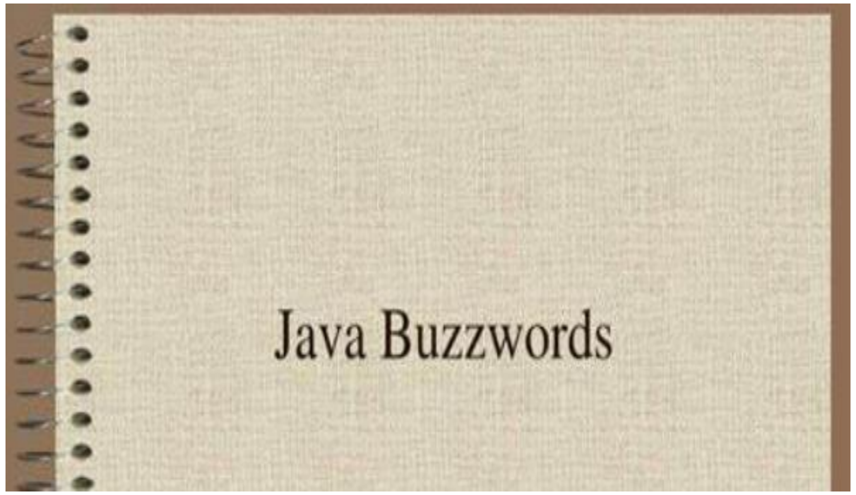
Java to learn Coding

1. it's Simple
2. Easy to read like English
3. Lots of learning material
4. Awesome help on Internet
5. Lots of libraries
6. Lots of code samples
7. Helps to get a job with good Pay.

RAMAIAH
Institute of Technology

According to Sun, Java is...

- Simple and Powerful
- Object Oriented
- Portable
- Architecture Neutral
- Distributed
- Multi-threaded
- Robust, Secure/Safe
- Interpreted
- High Performance
- Dynamic programming language/platform.



RAMAIAH Institute of Technology

As a whole, Java is a Comprehensive Programming Solution

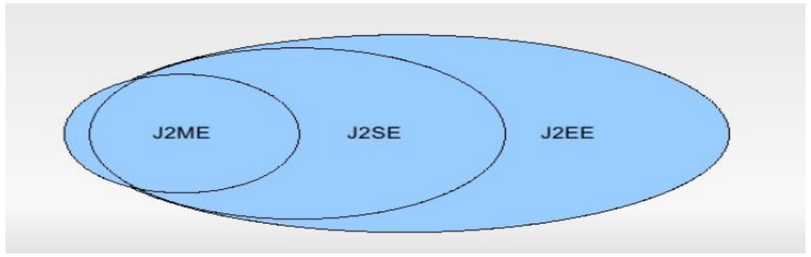
- Object Oriented
- Portable
- High Performance
- Geared for Distributed Environments
- Secure
- Multithreading
- Dynamic

RAMAIAH
Institute of Technology

JDK Versions

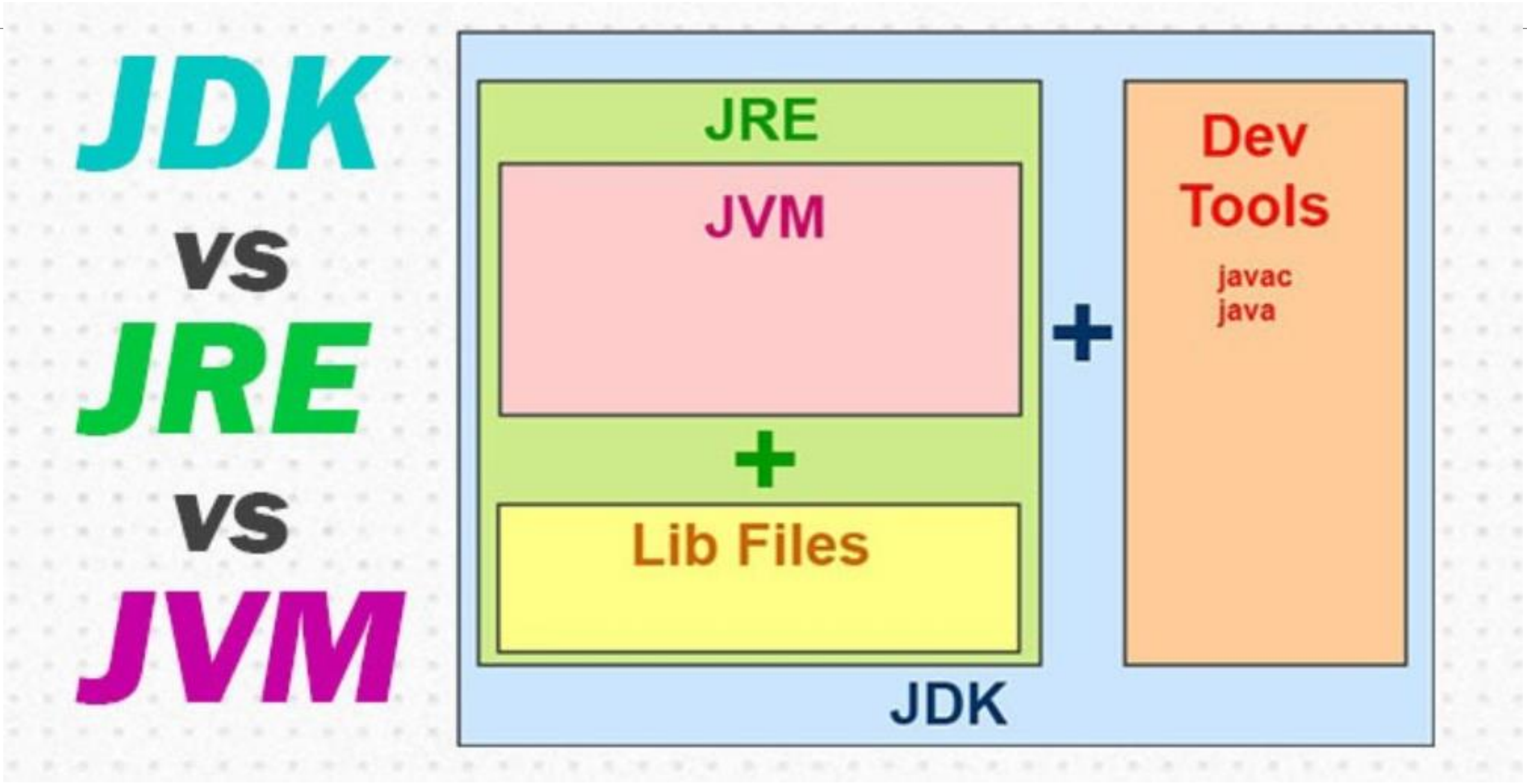
Release	Year
JDK Beta	1995
JDK 1.0	1996
JDK 1.1	1997
J2SE 1.2	1998
J2SE 1.3	2000
J2SE 1.4	2002
J2SE 5.0	2004
Java SE 6	2006
Java SE 7	2011
Java SE 8	2014

JDK Editions



- Java Standard Edition (J2SE)
  - J2SE can be used to develop client-side standalone applications or applets.
- Java Enterprise Edition (J2EE)
  - J2EE can be used to develop server-side applications such as Java servlets and Java ServerPages.
- Java Micro Edition (J2ME).
  - J2ME can be used to develop applications for mobile devices such as cell phones.

JDK vs JRE vs JVM



JDK vs JRE vs JVM

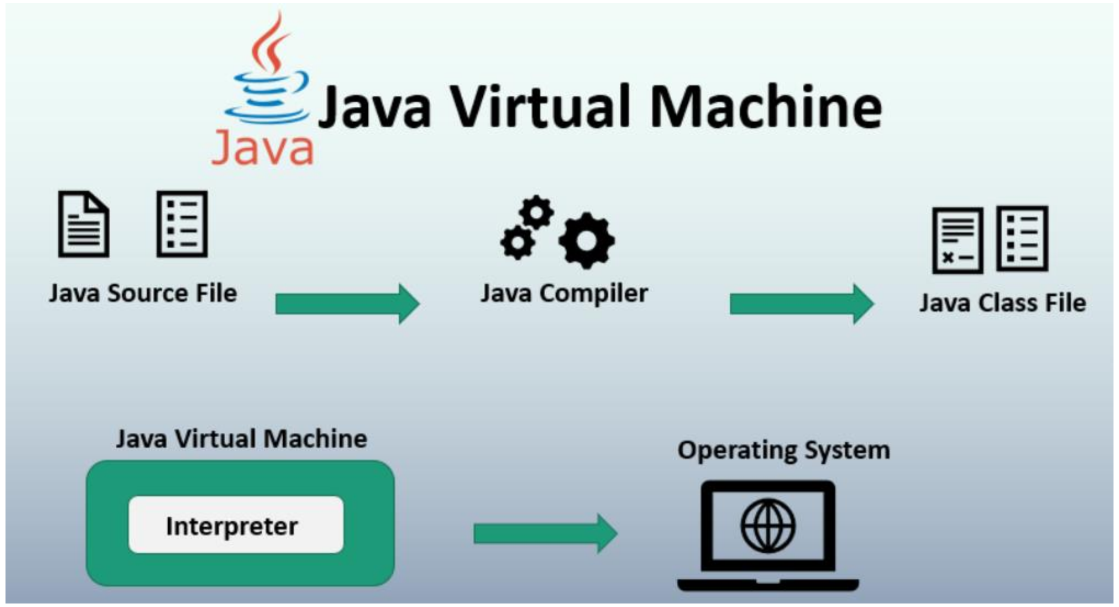
- **JDK (Java Development Kit)**
  - JDK contains everything that will be required to ***develop and run*** Java application.
- **JRE (Java Run time Environment)**
  - JRE contains everything required to ***run*** Java application which has already been compiled. It doesn’t contain the code library required to develop Java application.
- **JVM (Java Virtual Machine)**
  - JVM is a virtual machine which work on top of your operating system to provide a recommended environment for your compiled Java code. JVM only works with bytecode. Hence you need to compile your Java application(.java) so that it can be converted to bytecode format (also known as the .class file). Which then will be used by JVM to run application.

JDK vs JRE vs JVM

- Java Development Kit (JDK) consists of Java Runtime Environment (JRE) along with tools to compile and debug Java code for developing Java applications. JRE consists of libraries, Java Virtual Machine (JVM), Java Plugin and Java Web Start to run Java applications. JRE as a stand-alone does not contain compilers and debugging tools.

JVM (Java Virtual Machine)

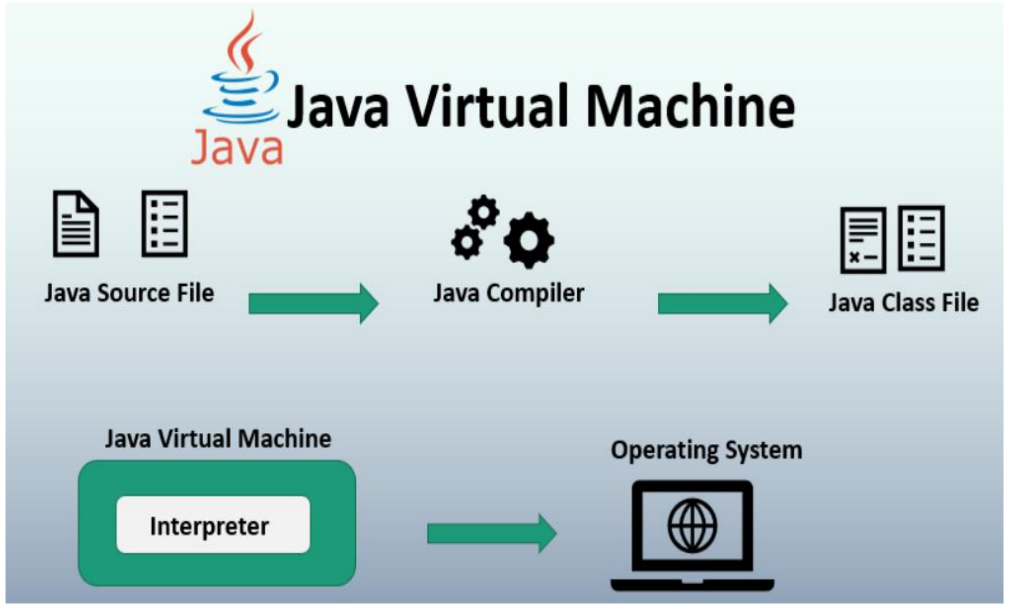
- JVM is a virtual machine or a program that provides run-time environment in which java byte code can be executed. JVMs are available for many hardware and software platforms. The use of the same byte code for all JVMs on all platforms make java platform independent.



RAMAIAH
Institute of Technology

JVM (Java Virtual Machine)

- ***Main task of JVM:***
  1. Search and locate the required files.
  2. Convert byte code into executable code.
  3. Allocate the memory into ram
  4. Execute the code.
  5. Delete the executable code

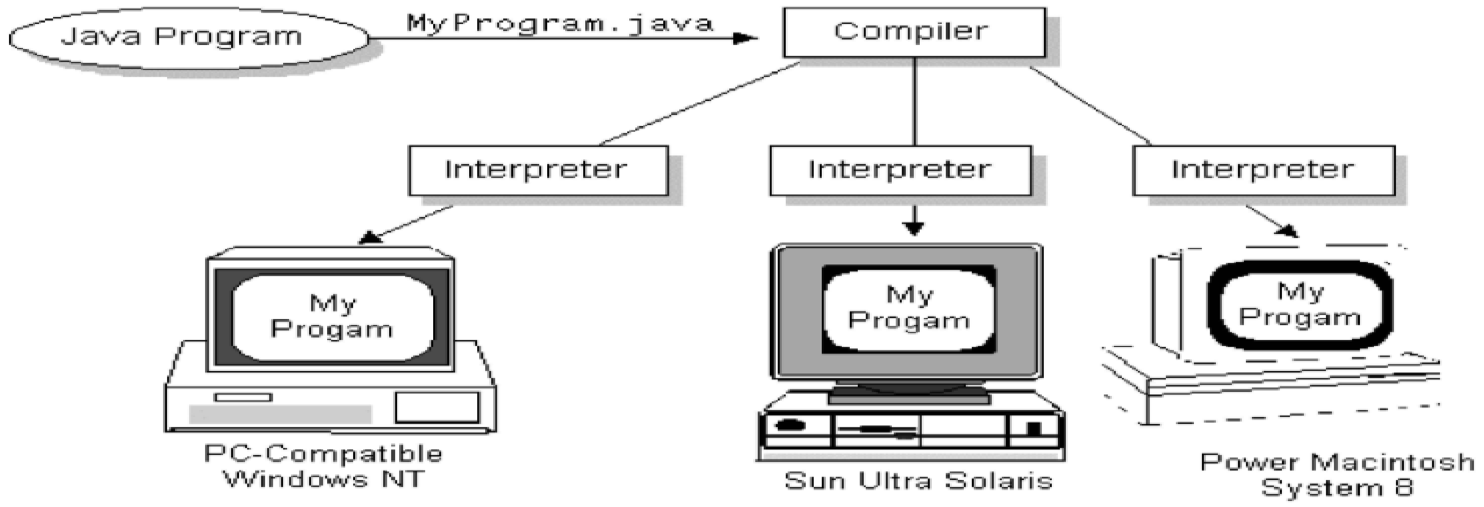


Java better than C++ ?

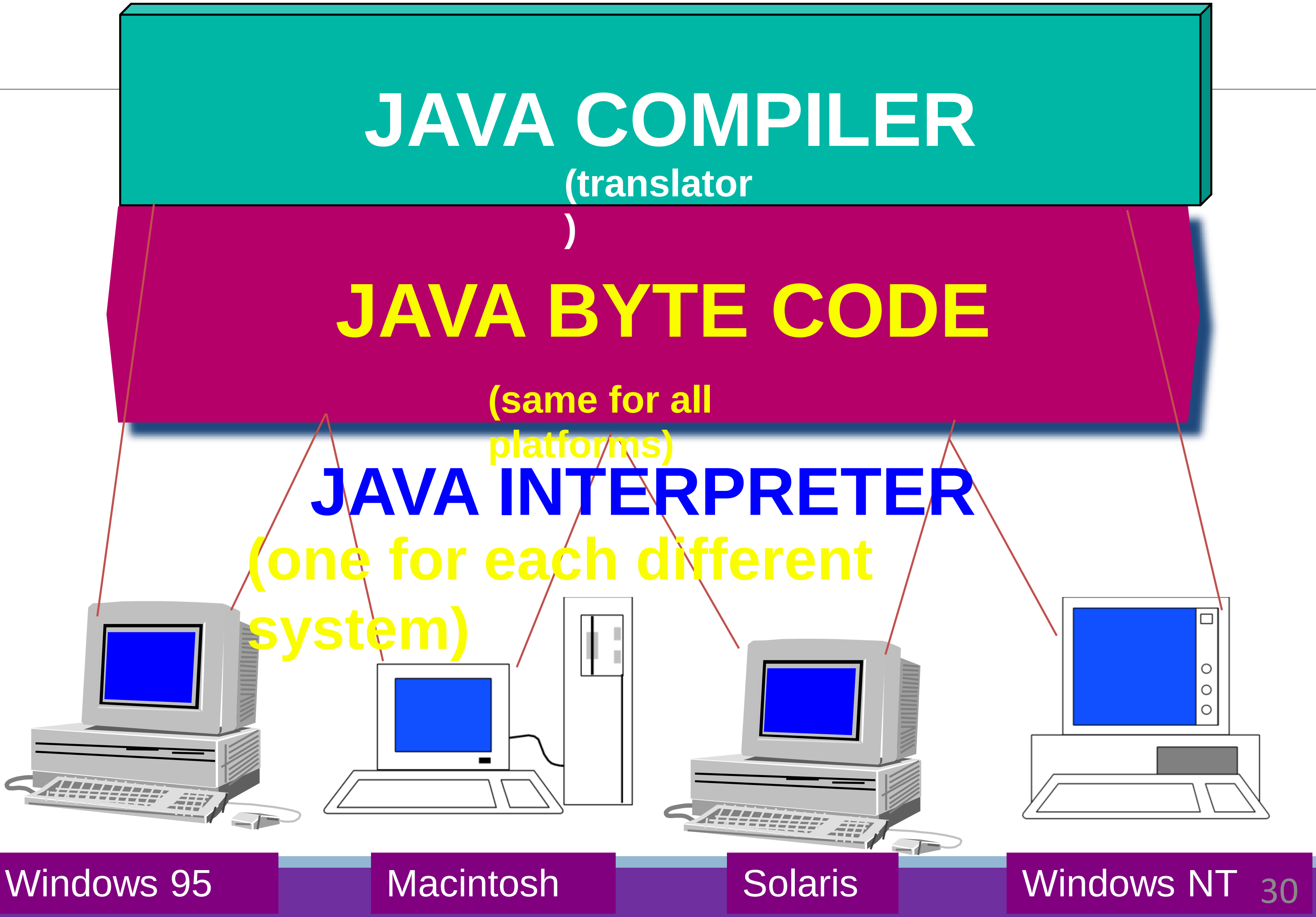
- No Typedefs, Defines, or Preprocessor
- No Global Variables
- No Goto statements
- No Pointers
- No Unsafe Structures
- No Multiple Inheritance
- No Operator Overloading
- No Automatic Coercions
- No Fragile Data Types

Base concept of java language

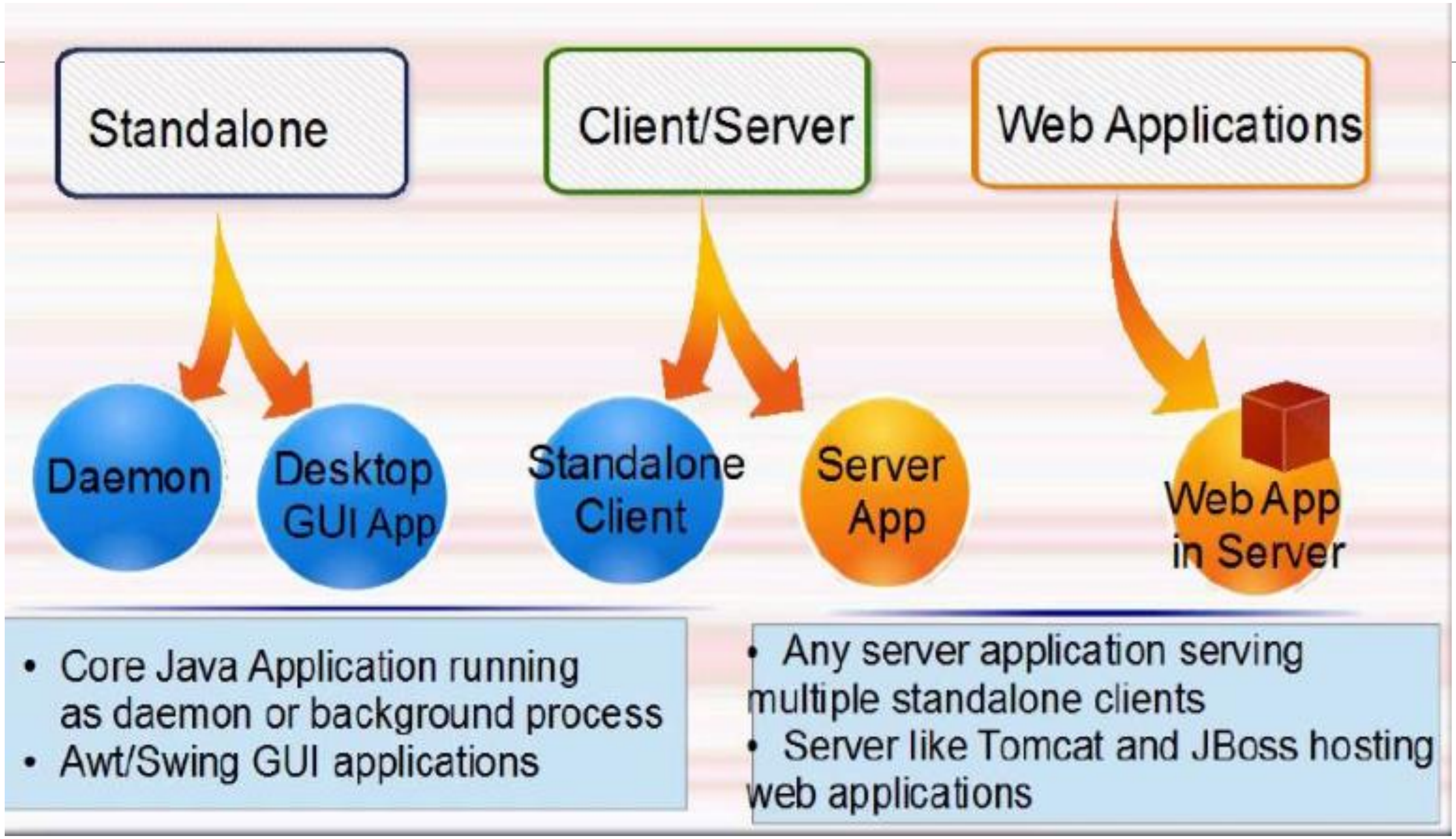
- **Write once, run anywhere (WORA)** – that means java program can run anywhere and on any platform. When java code is compiled it is converted into byte code. Now only this byte code is needed to run using JVM, no need of source code and recompilation.



Total Platform Independence



Types of Java Applications



- Core Java Application running as daemon or background process
- Awt/Swing GUI applications

- Any server application serving multiple standalone clients
- Server like Tomcat and JBoss hosting web applications

RAMAIAH Institute of Technology

Types of Java Applications

1) Standalone Application

- It is also known as desktop application or window-based application. An application that we need to install on every machine such as media player, antivirus etc. AWT and Swing are used in java for creating standalone applications.

2) Web Application

- An application that runs on the server side and creates dynamic page, is called web application. Currently, servlet, jsp, struts, jsf etc. technologies are used for creating web applications in java.

3) Enterprise Application

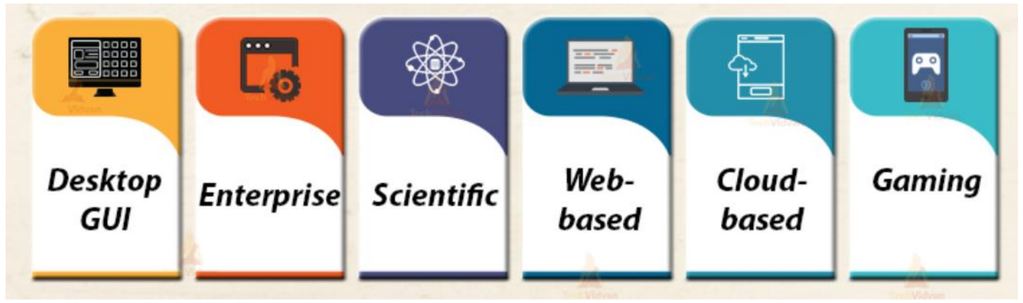
- An application that is distributed in nature, such as banking applications etc. It has the advantage of high level security, load balancing and clustering. In java, EJB is used for creating enterprise applications.

4) Mobile Application

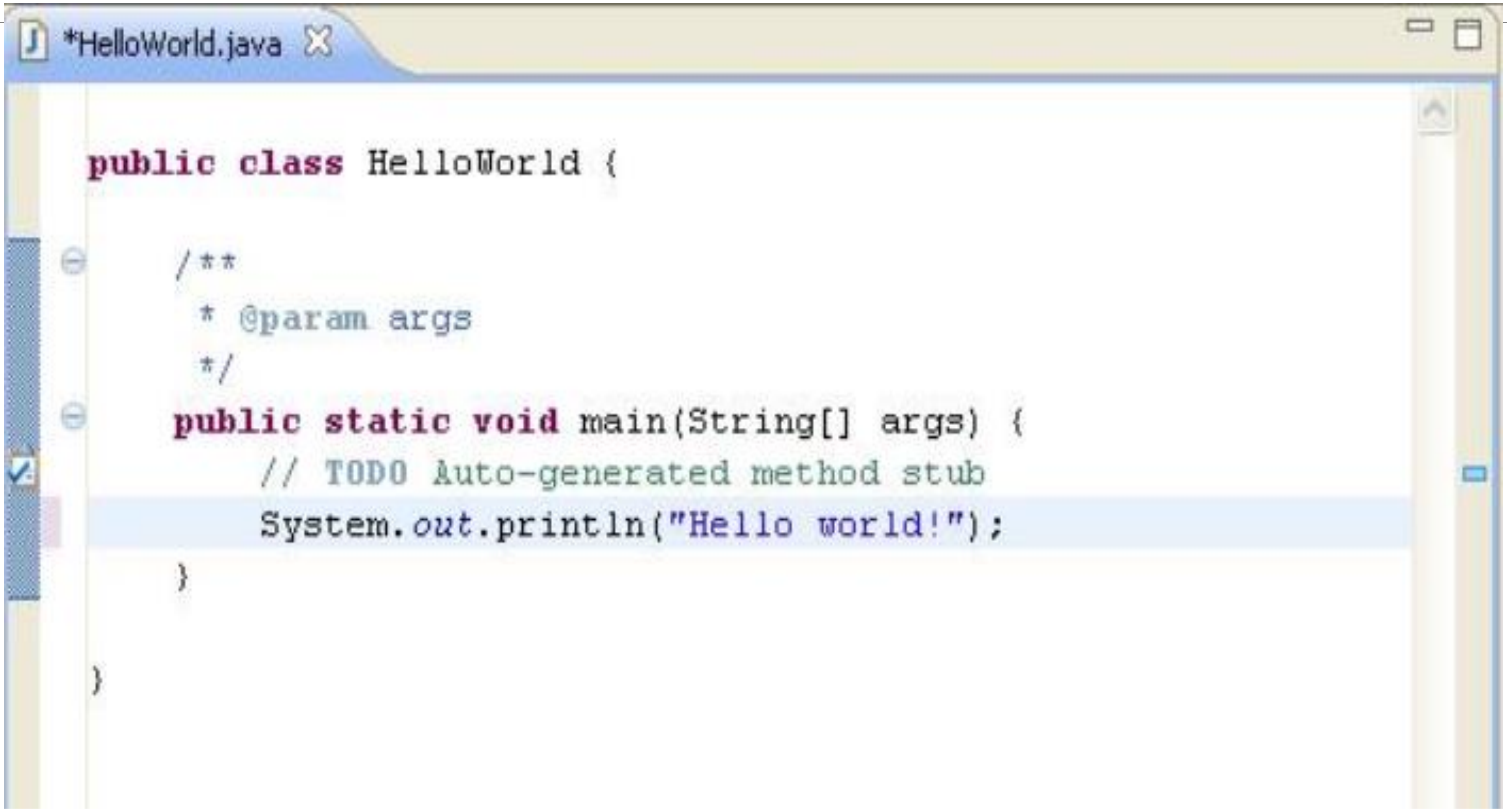
- An application that is created for mobile devices. Currently Android and Java ME are used for

Application of Java

- Java is widely used in every corner of world and of human life. Java is not only used in softwares but is also widely used in designing hardware controlling software components. There are more than 930 million JRE downloads each year and 3 billion mobile phones run java.
- Following are some other usage of Java :
- Developing Desktop Applications
- Web Applications like Linkedin.com, Snapdeal.com etc
- Mobile Operating System like Android
- Embedded Systems
- Robotics and games etc.



First Java Program



*HelloWorld.java

```
public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello world!");
    }

}
```

HelloWorld.java

First Java Program

The keyword '**public**' says that, the main function can be accessed even outside the defining class.

The keyword '**static**' indicates that a function can be accessed without an object.

The keyword '**void**' indicates that a function returns nothing.

The Keyword '**main**' is the name of the function and entry point for a Java application.

Variants of main() in Java

Default:

- **public static void main(String args[])**

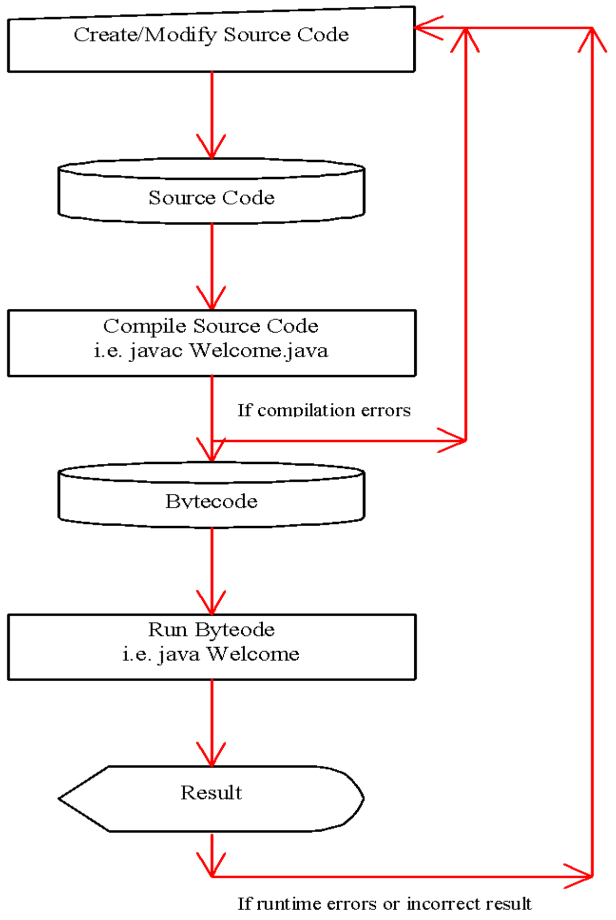
Variants:

- **static** public void main(String args[])
- public static void **main(String[] args**)
- public **final** static void main(String[] args)
- public **synchronized** static void main(String[] args)

The main() can be overloaded but not overridden.

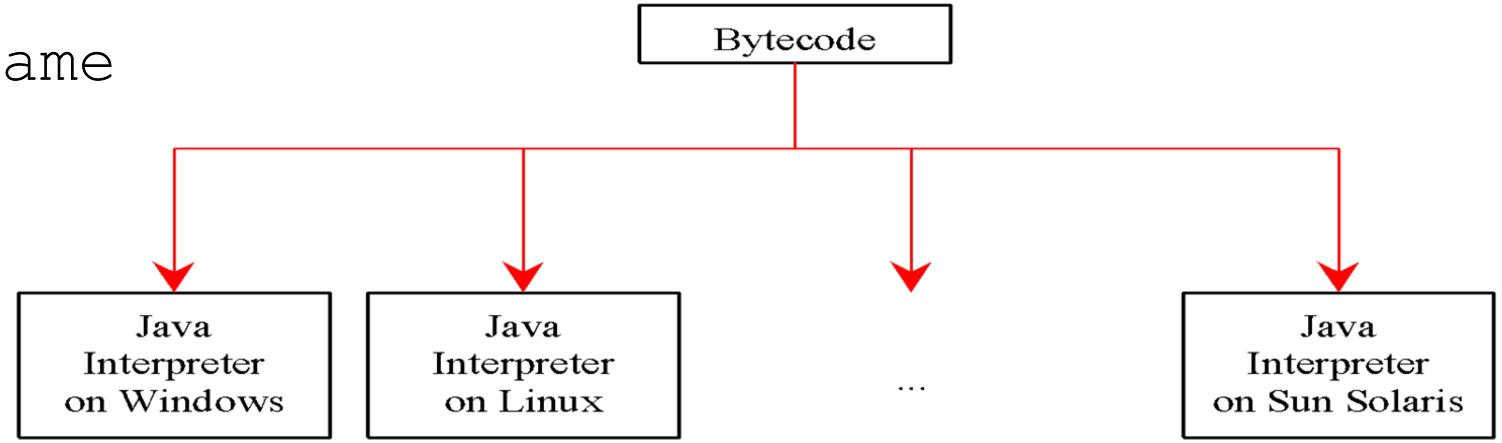
Creating and Compiling Programs

- On command line
  - `javac file.java`



Executing programs

- On command line
  - `java classname`


Bytecode
Java Interpreter on Windows
Java Interpreter on Linux
...
Java Interpreter on Sun Solaris

```
javac Welcome.java
java Welcome
output:...
```

RAMAIAH
Institute of Technology

Java Installation

- Is java already installed

```
java –version
```

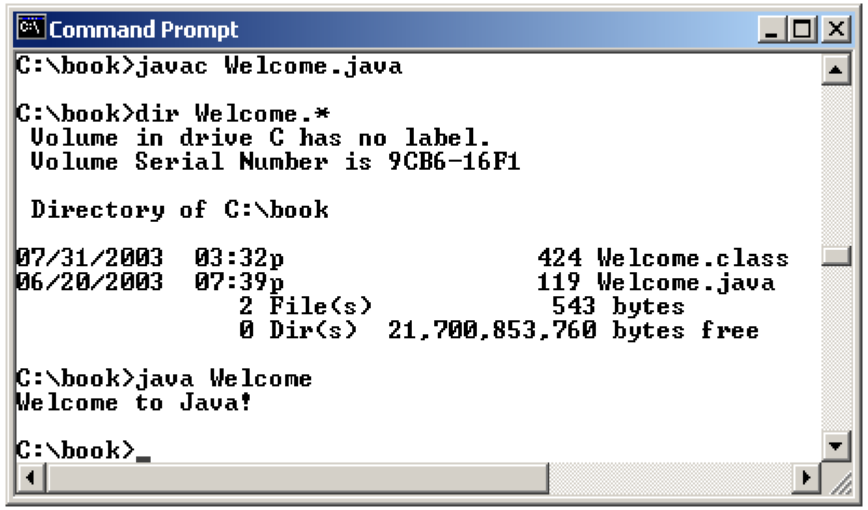
- To install java on ubuntu run the following commands sequentially.

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:sun-java-community-team/sun-java6
sudo apt-get update
sudo apt-get install sun-java6-jdk
```

RAMAIAH Institute of Technology

Compiling and Running Java from the Command Window

- Set path to JDK bin directory
  - set path=c:\Program Files\java\jdk1.6.0\bin
- Compile
  - javac Welcome.java
- Run
  - java Welcome


Command Prompt

```
C:\book>javac Welcome.java

C:\book>dir Welcome.*
 Volume in drive C has no label.
 Volume Serial Number is 9CB6-16F1

 Directory of C:\book

07/31/2003  03:32p                 424 Welcome.class
06/20/2003  07:39p                 119 Welcome.java
               2 File(s)            543 bytes
               0 Dir(s)  21,700,853,760 bytes free

C:\book>java Welcome
Welcome to Java!

C:\book>_
```

Popular Java Editors

- To write your Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market. But for now, you can consider one of the following:
- **Notepad:** On Windows machine you can use any simple text editor like Notepad, TextPad. On Ubuntu you can use Gedit or VIM
- **Netbeans:** is a Java IDE that is open-source and free which can be downloaded
- **Eclipse:** is also a Java IDE developed by the eclipse open-source community and can be downloaded.

Java Coding Guidelines

- About Java programs, it is very important to keep in mind the following points.
- **Case Sensitivity -** Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java.
- **Class Names -** For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case. Eg. *class MyFirstJavaClass*
- **Method Names -** All method names should start with a Lower Case letter. Eg. *public void myMethodName()*
- **Program File Name -** Name of the program file should exactly match the class name. When saving the file, you should save it using the class name and append '.java' to the end of the name. If the file name and the class name do not match your program will not compile.
  Example: Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as *'MyFirstJavaProgram.java'*
- **public static void main(String args[]) -** Java program processing starts from the main() method which is a mandatory part of every Java program

Java Coding Guidelines

Naming variable:

- Use first word in small letters and all remaining words will be capitalized. E.g. – rollNumber, firstName.

Naming Constants:

- Use all letters in upper case. E.g. – MAX_MARKS**.**

Comment:

- For clarity of the code add comments.

Comments in Java

Three types of comments in Java

- *Line comment: A line comment is preceded by two slashes (//) in a line.*
- *Paragraph comment: A paragraph comment is enclosed between /* and */ in one or multiple lines.*
- *javadoc comment*: javadoc comments begin with /** and end with */. They are used for documenting classes, data, and methods. They can be extracted into an HTML file using JDK's javadoc command.

- Java Identifiers:
  - All Java components require names. Names used for classes, variables and methods are called identifiers.
- Java Modifiers:
  - **Access Modifiers:** default, public , protected, private
  - **Non-access Modifiers:** final, abstract, strictfp
- Java Variables:
  - Local Variables
  - Class Variables (Static Variables)
  - Instance Variables (Non-static variables)

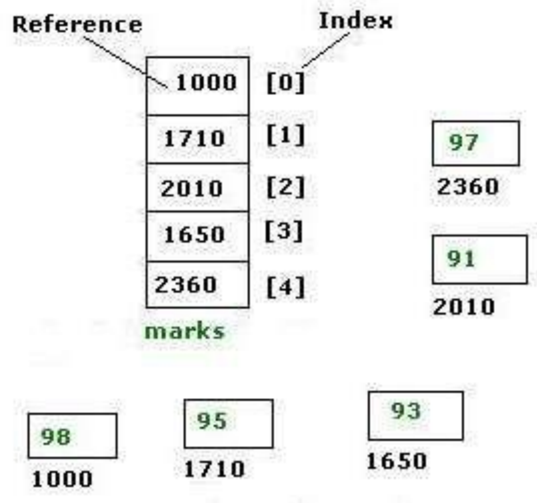
JAVA ARRAYS AND CONTROL STATEMENTS

Java
Programming

Java Arrays

- In java arrays are objects. All methods of an Object can be invoked on an array. Arrays are stored in heap memory.

Physical view of an Array



- **Array Declaration**

//Single Dimensional Array

int[] arr; *//recommended*

int arr[];

//Multi Dimensional Array

int[][] arr; *//recommended*

int arr[][];

int[] arr[];

- **Array Instantiation**

marks = new int[5];

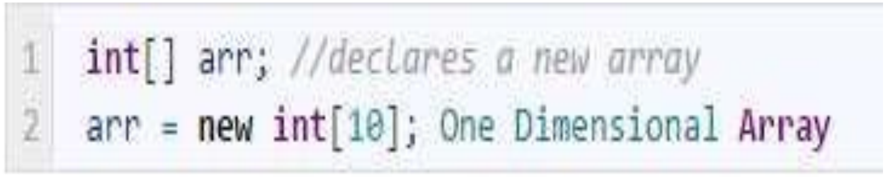
- **Java array initialization and instantiation together**

int marks[] = {98, 95, 91, 93, 97};

Constructing an Java Array

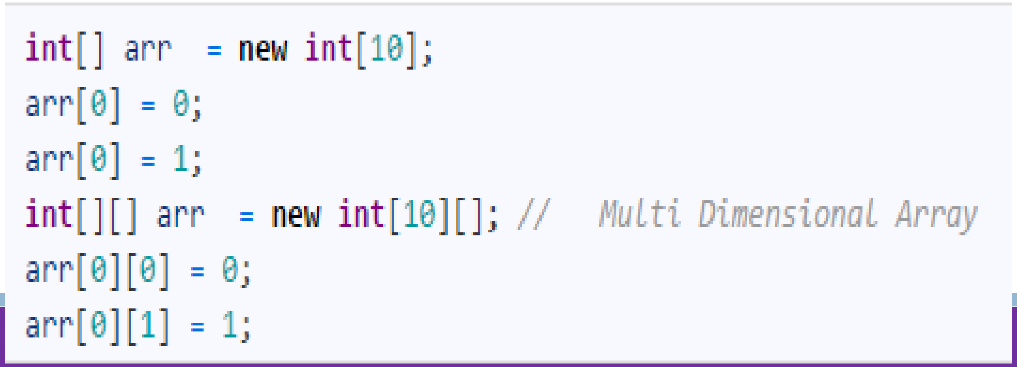
One Dimensional Array

- New keyword will be used to construct one/multi dimensional array.

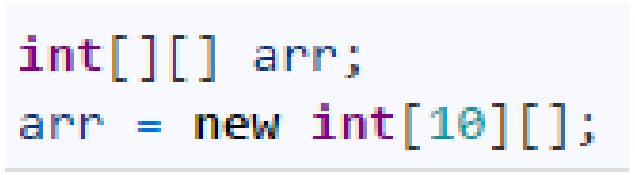


```
int[] arr; //declares a new array
arr = new int[10]; One Dimensional Array
```

- **Two Dimensional Array**
- These are array of arrays. So a two dimensional array is array of arrays of int
- **Initializing Array**



```
int[] arr  = new int[10];
arr[0] = 0;
arr[0] = 1;
int[][] arr  = new int[10][]; //   Multi Dimensional Array
arr[0][0] = 0;
arr[0][1] = 1;
```

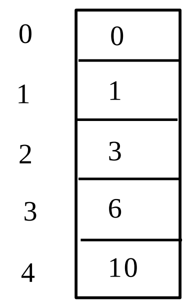


```
int[][] arr;
arr = new int[10][];
```

Iterating a Java Array

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
    values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the fourth iteration



TestArray.java

Initializing arrays with input values

```
myList.length=5;
java.util.Scanner input = new java.util.Scanner(System.in);
System.out.print("Enter " + myList.length + "
      values: ");
for (int i = 0; i < myList.length; i++)
  myList[i] = input.nextDouble();
```

- Initializing arrays with random values

```
for (int i = 0; i < myList.length; i++)
{
  myList[i] = Math.random() * 100;
}
```

Declare/Create Two-dimensional Arrays

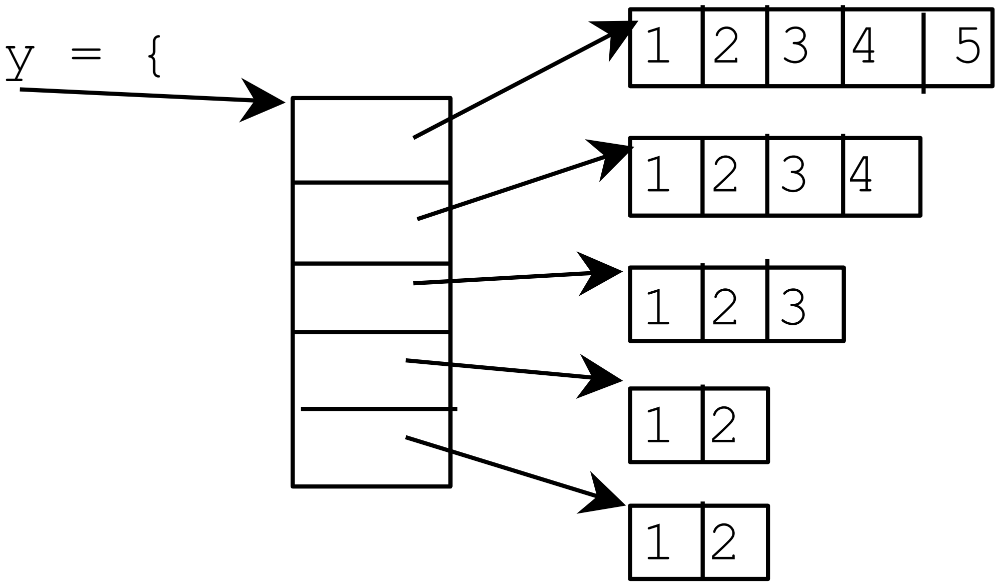
```
// Declare array ref var
dataType[][] refVar;

// Create array and assign its reference to variable
refVar = new dataType[10][10];

// Combine declaration and creation in one statement
dataType[][] refVar = new dataType[10][10];

// Alternative syntax
dataType refVar[][] = new dataType[10][10];
```

RAMAIAH Institute of Technology

Ragged Arrays

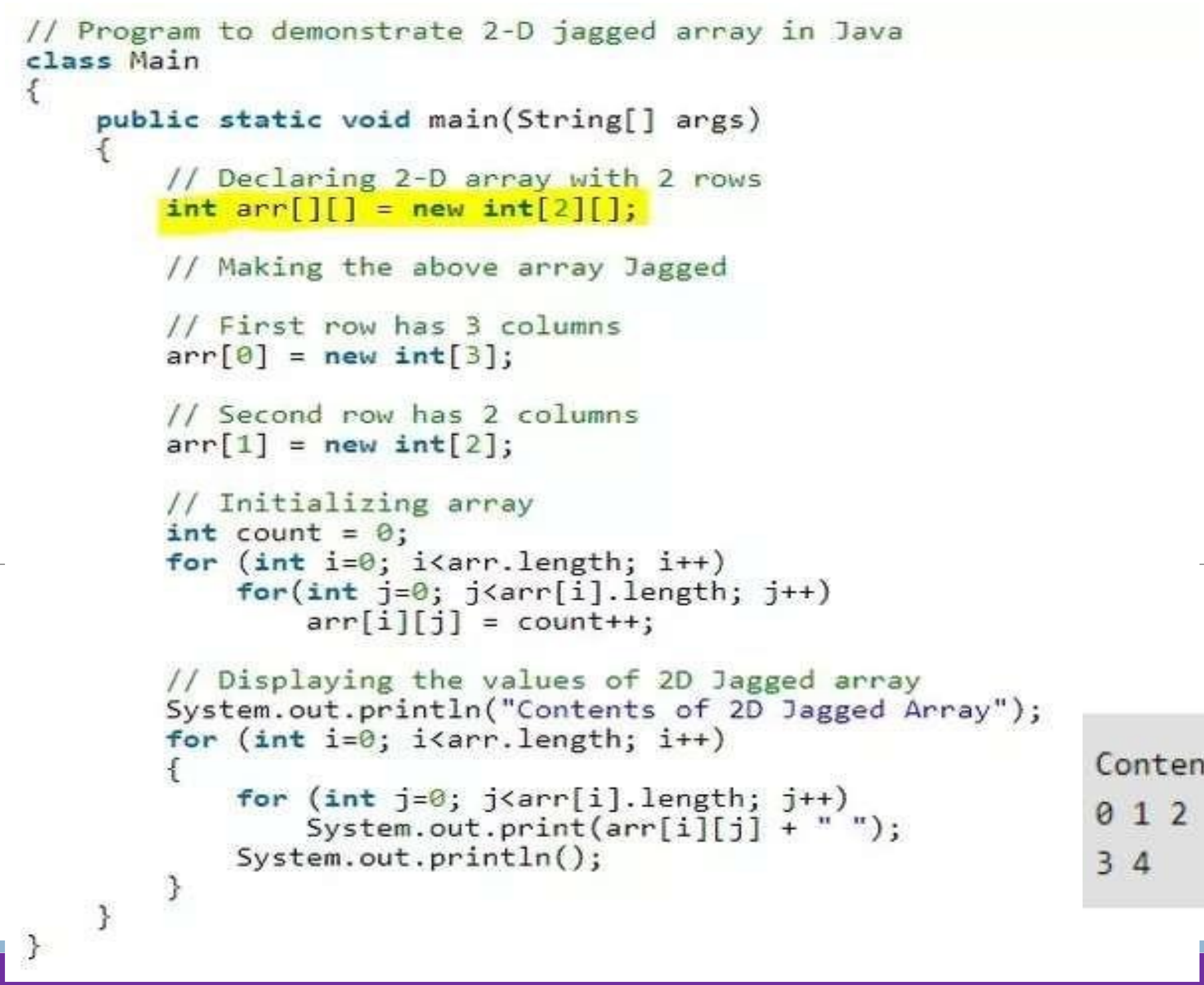
Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as *a ragged array*. For example,

```
int[][] triangleArray = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```



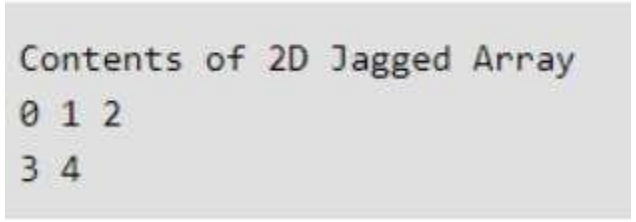
RAMAIAH
Institute of Technology

Example



```
// Program to demonstrate 2-D jagged array in Java
class Main
{
    public static void main(String[] args)
    {
        // Declaring 2-D array with 2 rows
        int arr[][] = new int[2][];

        // Making the above array Jagged

        // First row has 3 columns
        arr[0] = new int[3];

        // Second row has 2 columns
        arr[1] = new int[2];

        // Initializing array
        int count = 0;
        for (int i=0; i<arr.length; i++)
            for(int j=0; j<arr[i].length; j++)
                arr[i][j] = count++;

        // Displaying the values of 2D Jagged array
        System.out.println("Contents of 2D Jagged Array");
        for (int i=0; i<arr.length; i++)
        {
            for (int j=0; j<arr[i].length; j++)
                System.out.print(arr[i][j] + " ");
            System.out.println();
        }
    }
}
```

Main.java



```
Contents of 2D Jagged Array
0 1 2
3 4
```

RAMAIAH Institute of Technology

Flow control

Basically, it is exactly like c/c++.

if/else

```
If(x==4) {
   // act1
} else {
   // act2
}
```

do/while

```
int i=5;
do {
   // act1
   i--;
} while(i!=0);
```

for

```
int j;
for(int i=0;i<=9;i++)
{
   j+=i;
}
```

switch

```
char
c=IN.getChar();
switch(c) {
   case 'a':
   case 'b':
      // act1
      break;
   default:
      // act2
}
```

If else statement

PositiveNegativeExample.java

LeapYearExample.java

JavaNestedIfExample2.java

JavaNestedIfExample.java

IfElseExample.java

IfExample.java

IfElseTernaryExample.java

IfElseIfExample.java

RAMAIAH
Institute of Technology

switch Statements

```
switch (status) {
  case 0:  compute taxes for single filers;
           break;
  case 1:  compute taxes for married file jointly;
           break;
  case 2:  compute taxes for married file separately;
           break;
  case 3:  compute taxes for head of household;
           break;
  default: System.out.println("Errors: invalid status");
           System.exit(0);
}
```

`switch` Statements

SwitchVowelExample.java

SwitchStringExample.java

SwitchExample.java

NestedSwitchExample.java

RAMAIAH
Institute of Technology

Flow control contd…

- break is used in the loops and when executed, the control of the execution will come out of the loop.
- continue makes the loop to skip the current execution and continues with the next iteration.
- return statement can be used to cause execution to branch back to the caller of the method.
- Labeled break and continue statements will break or continue from the loop that is mentioned. Used in nested loops.

For loop

NestedForExample.java

LabeledForExample2.java

PyramidExample.java

LabeledForExample.java

ForExample.java

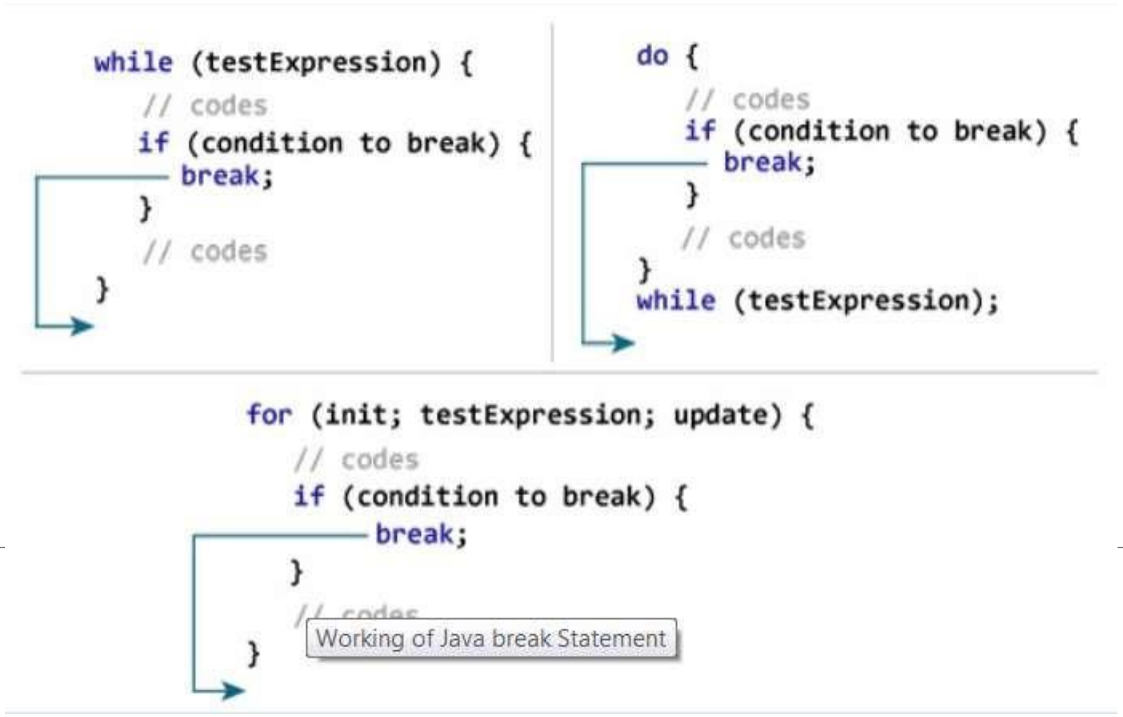
While and do while

DoWhileExample.java

WhileExample.java

RAMAIAH
Institute of Technology

Break statement



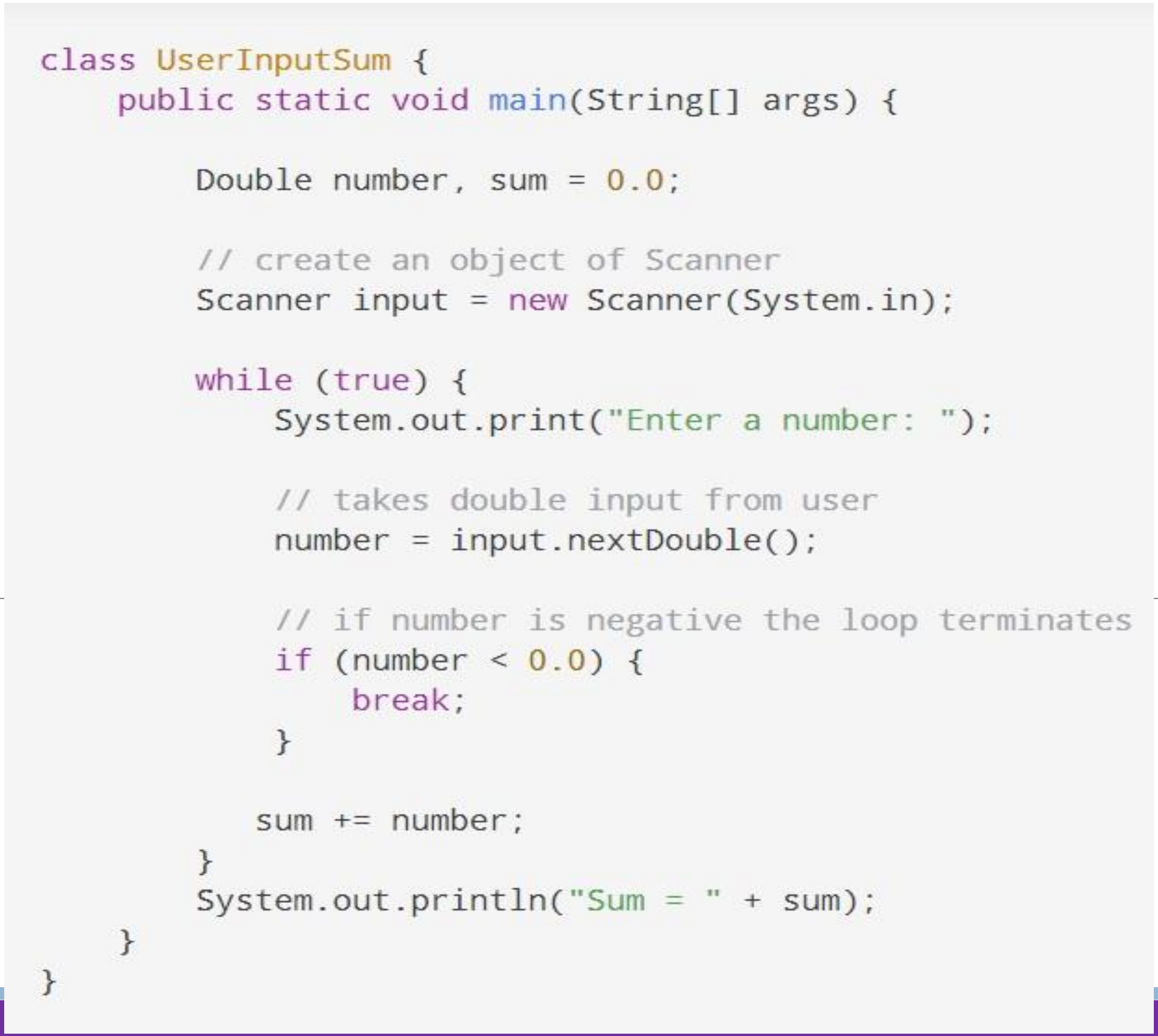
```
while (testExpression) {
    // codes
    if (condition to break) {
      break;
    }
    // codes
}
```

```
do {
    // codes
    if (condition to break) {
      break;
    }
    // codes
}
while (testExpression);
```

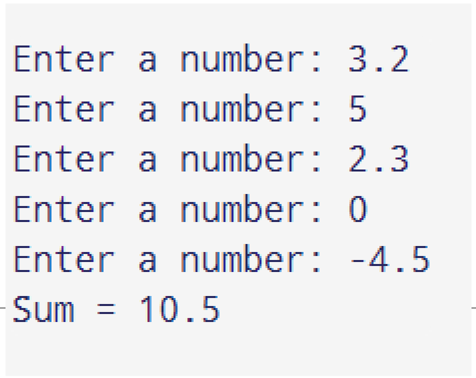
```
for (init; testExpression; update) {
    // codes
    if (condition to break) {
      break;
    }
    // codes
}
```

Working of Java break Statement

Example

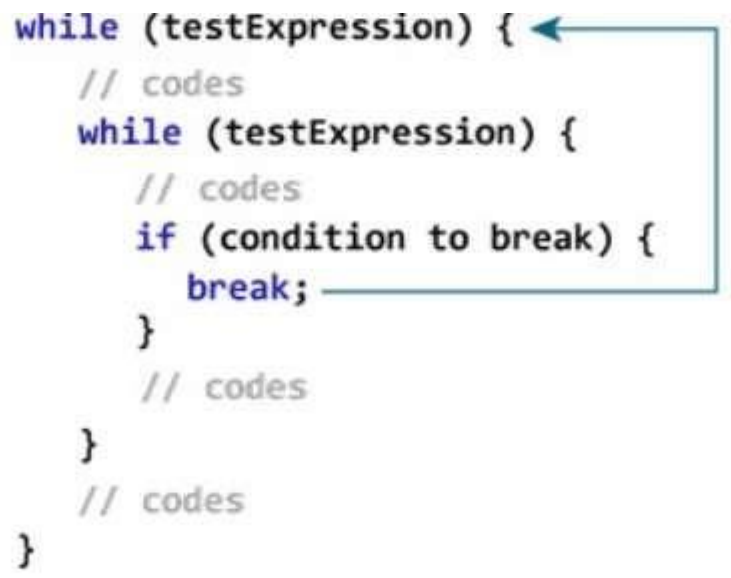


```
class UserInputSum {
    public static void main(String[] args) {

        Double number, sum = 0.0;

        // create an object of Scanner
        Scanner input = new Scanner(System.in);

        while (true) {
            System.out.print("Enter a number: ");

            // takes double input from user
            number = input.nextDouble();

            // if number is negative the loop terminates
            if (number < 0.0) {
                break;
            }

           sum += number;
        }
        System.out.println("Sum = " + sum);
    }
}
```



```
Enter a number: 3.2
Enter a number: 5
Enter a number: 2.3
Enter a number: 0
Enter a number: -4.5
Sum = 10.5
```

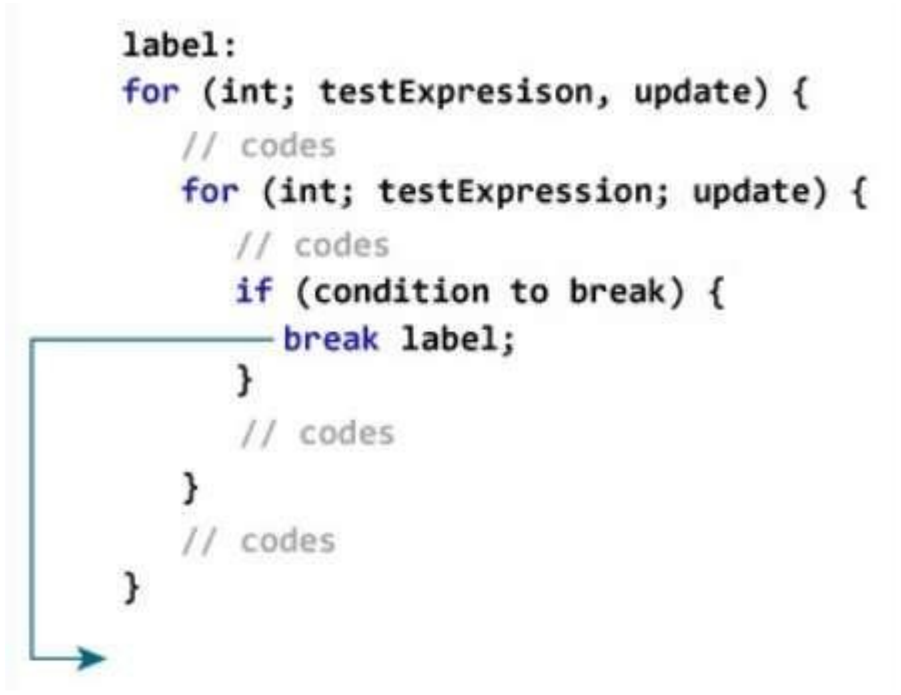
Java break and Nested Loop



```
while (testExpression) {
   // codes
   while (testExpression) {
      // codes
      if (condition to break) {
         break;
      }
      // codes
   }
   // codes
}
```

Labeled break Statement

- We can use the labeled break statement to terminate the outermost loop.

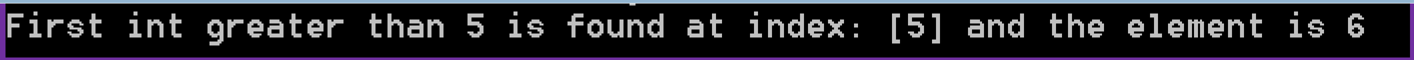


```
label:
for (int; testExpresison, update) {
    // codes
    for (int; testExpression; update) {
       // codes
       if (condition to break) {
          break label;
       }
       // codes
    }
    // codes
}
```

RAMAIAH
Institute of Technology

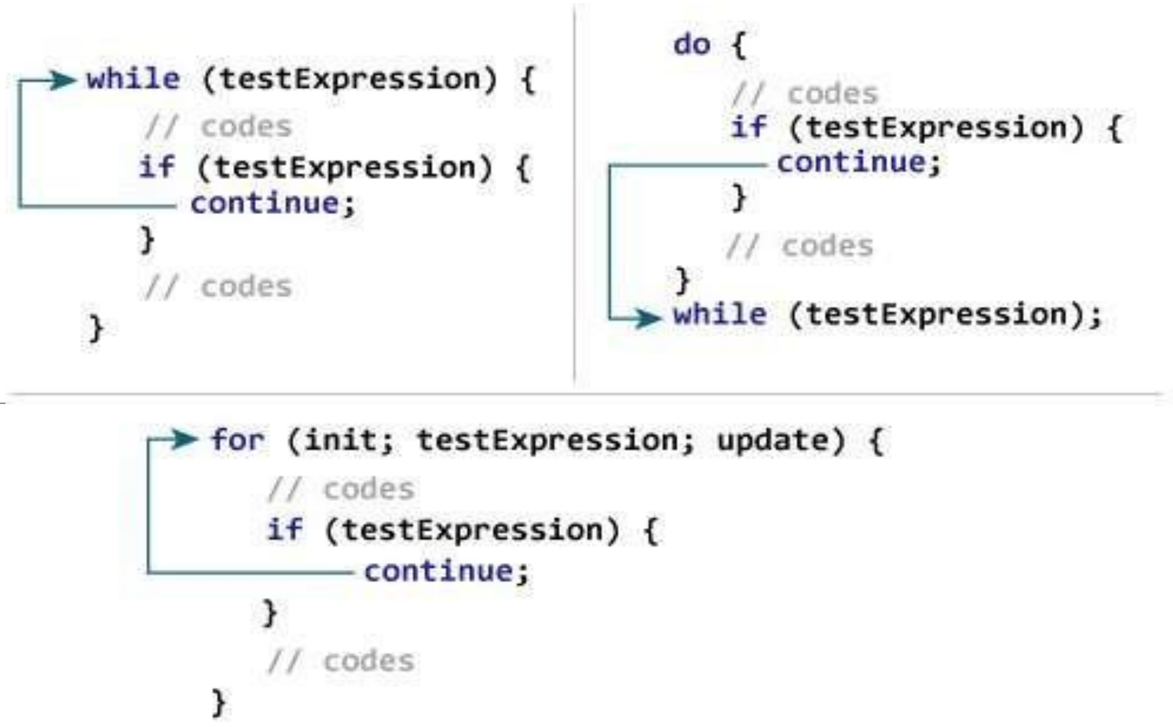
Java break statement with label example

```
public class JavaBreakLabel {
public static void main(String[] args) {
    int[]arr = { 1,2,3,4,5,6,7,6,8,9,10 };
    boolean found = false;
    int row = 0;
    //find index of first int greater than 5
    searchint:
    for (row = 0; row < arr.length; row++) {
                    if (arr[row] > 5) {
                            found = true;
                            //using break label to terminate outer statements
                            break searchint;
                    }
    }
    if (found)
  System.out.println("First int greater than 5 is found at index: [" + row + "] and the
element is "+arr[row]);
}
}
```



```
First int greater than 5 is found at index: [5] and the element is 6
```

Java continue statement



```
while (testExpression) {
    // codes
    if (testExpression) {
       continue;
    }
    // codes
}
```

```
do {
    // codes
    if (testExpression) {
       continue;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (testExpression) {
       continue;
    }
    // codes
}
```

BreakWhileExample.java

BreakExample2.java

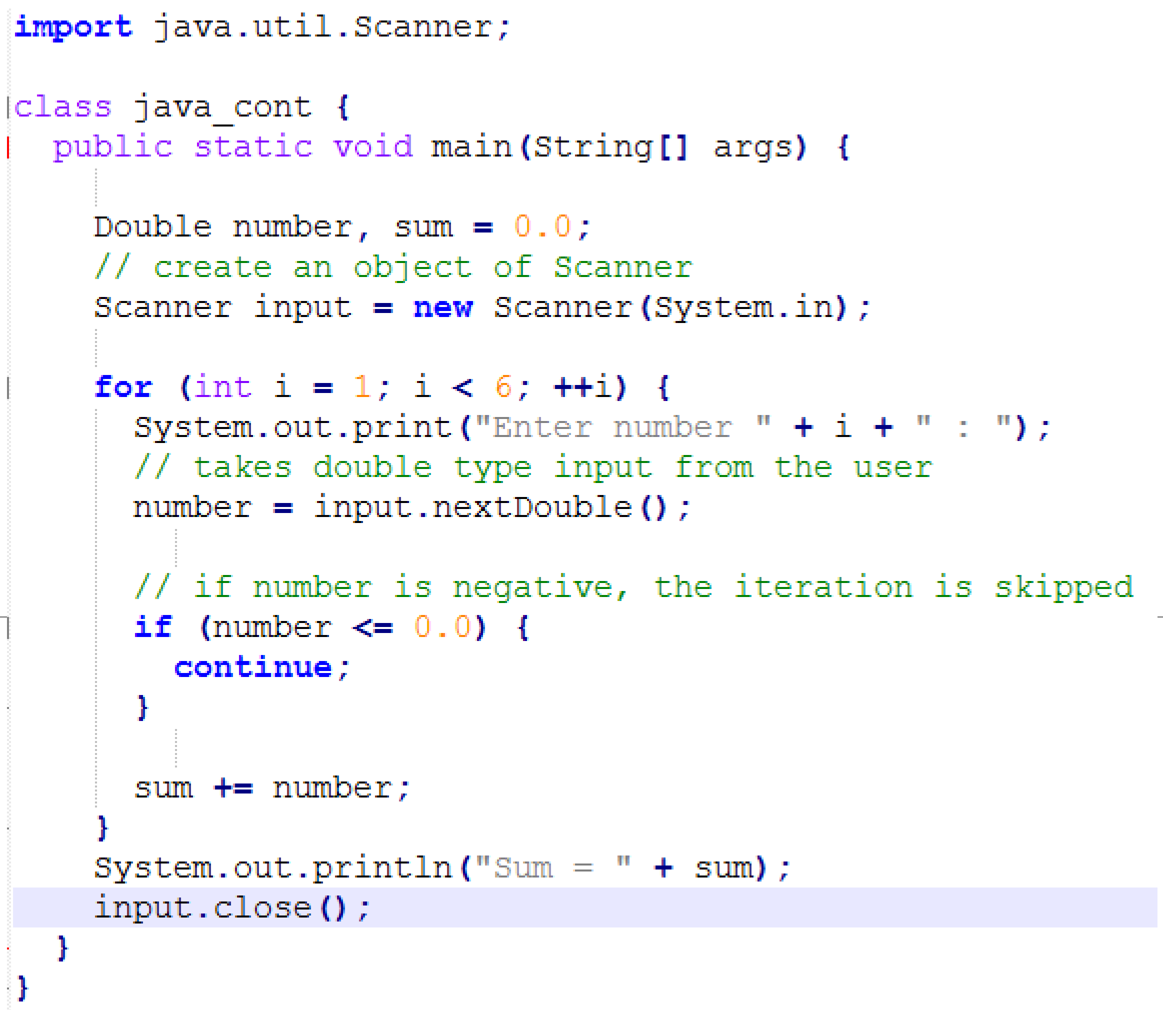
BreakDoWhileExample.java

BreakExample3.java

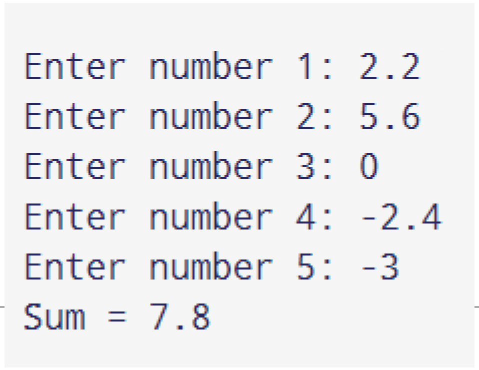
BreakExample.java

RAMAIAH Institute of Technology

Example

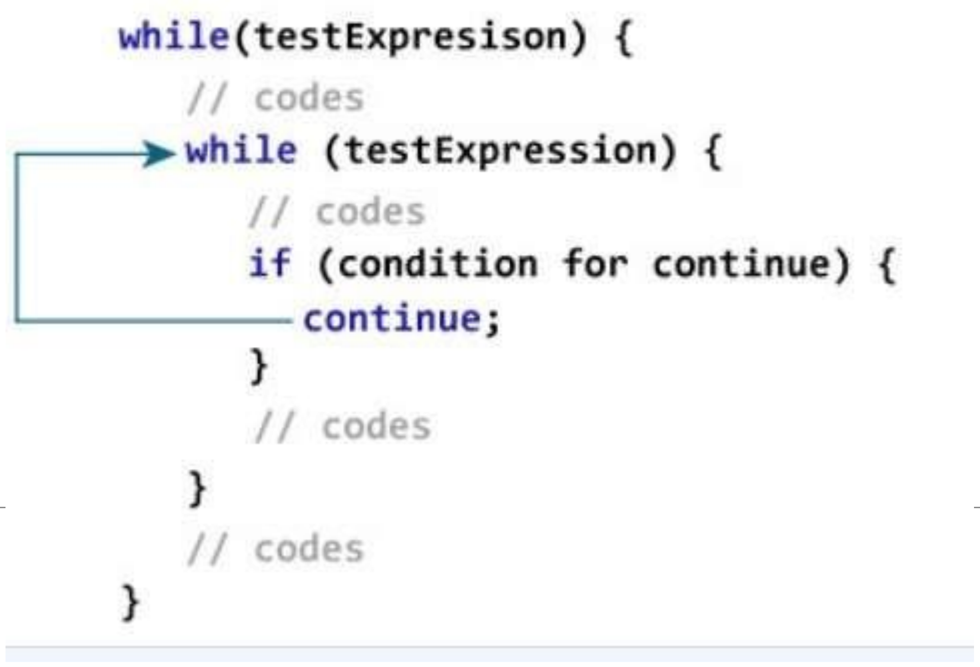


```java
import java.util.Scanner;

class java_cont {
  public static void main(String[] args) {

    Double number, sum = 0.0;
    // create an object of Scanner
    Scanner input = new Scanner(System.in);

    for (int i = 1; i < 6; ++i) {
      System.out.print("Enter number " + i + " : ");
      // takes double type input from the user
      number = input.nextDouble();

      // if number is negative, the iteration is skipped
      if (number <= 0.0) {
        continue;
      }

      sum += number;
    }
    System.out.println("Sum = " + sum);
    input.close();
  }
}
```



```
Enter number 1: 2.2
Enter number 2: 5.6
Enter number 3: 0
Enter number 4: -2.4
Enter number 5: -3
Sum = 7.8
```

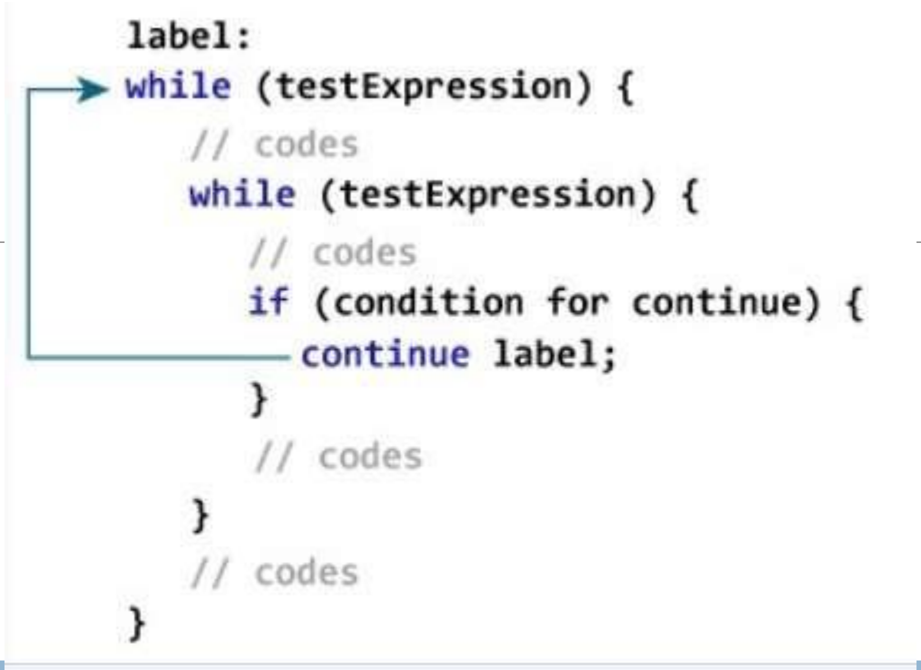
Java continue and Nested Loop



```
while(testExpresison) {
    // codes
    while (testExpression) {
        // codes
        if (condition for continue) {
            continue;
        }
        // codes
    }
    // codes
}
```

Labeled continue Statement

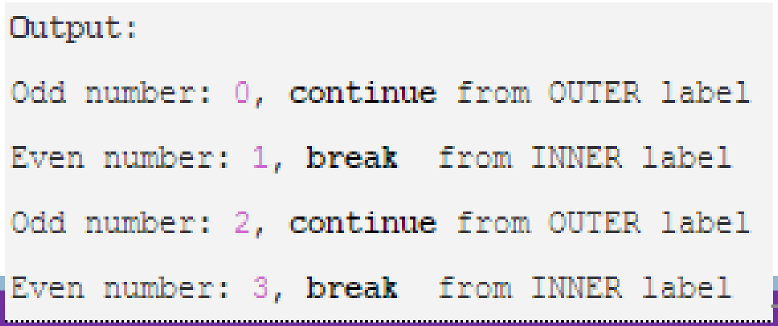
- It is used to terminate the innermost loop and switch statement. However, there is another form of continue statement in Java known as labeled continue.



```
label:
while (testExpression) {
   // codes
   while (testExpression) {
      // codes
      if (condition for continue) {
         continue label;
      }
      // codes
   }
   // codes
}
```

Java continue statement with label example

```
public class BreakContinueWithLabel {
     public static void main(String args[]) {
          int[] numbers= new int[]{100,18,21,30};
     //Outer loop checks if number is multiple of 2
          OUTER:  //outer label
          for(int i = 0; i<numbers.length; i++)
            {
             if(i % 2 == 0)
              {
                 System.out.println("Odd number: " + i + ", continue from OUTER label");
                 continue OUTER;
             }
             INNER:
             for(int j = 0; j<numbers.length; j++)
             {
                 System.out.println("Even number: " + i + ", break  from INNER label");
                 break INNER;
             }
          }

      }
   }
```



```
Output:

Odd number: 0, continue from OUTER label

Even number: 1, break  from INNER label

Odd number: 2, continue from OUTER label

Even number: 3, break  from INNER label
```

Example

```
class LabeledContinue {
    public static void main(String[] args) {

        // the outer for loop is labeled as label
        first:
        for (int i = 1; i < 6; ++i) {
            for (int j = 1; j < 5; ++j) {
                if (i == 3 || j == 2)

                    // skips the iteration of label (outer for loop)
                    continue first;
                System.out.println("i = " + i + "; j = " + j);
            }
        }
    }
}
```

```
i = 1; j = 1
i = 2; j = 1
i = 4; j = 1
i = 5; j = 1
```

ContinueWhileExample.java

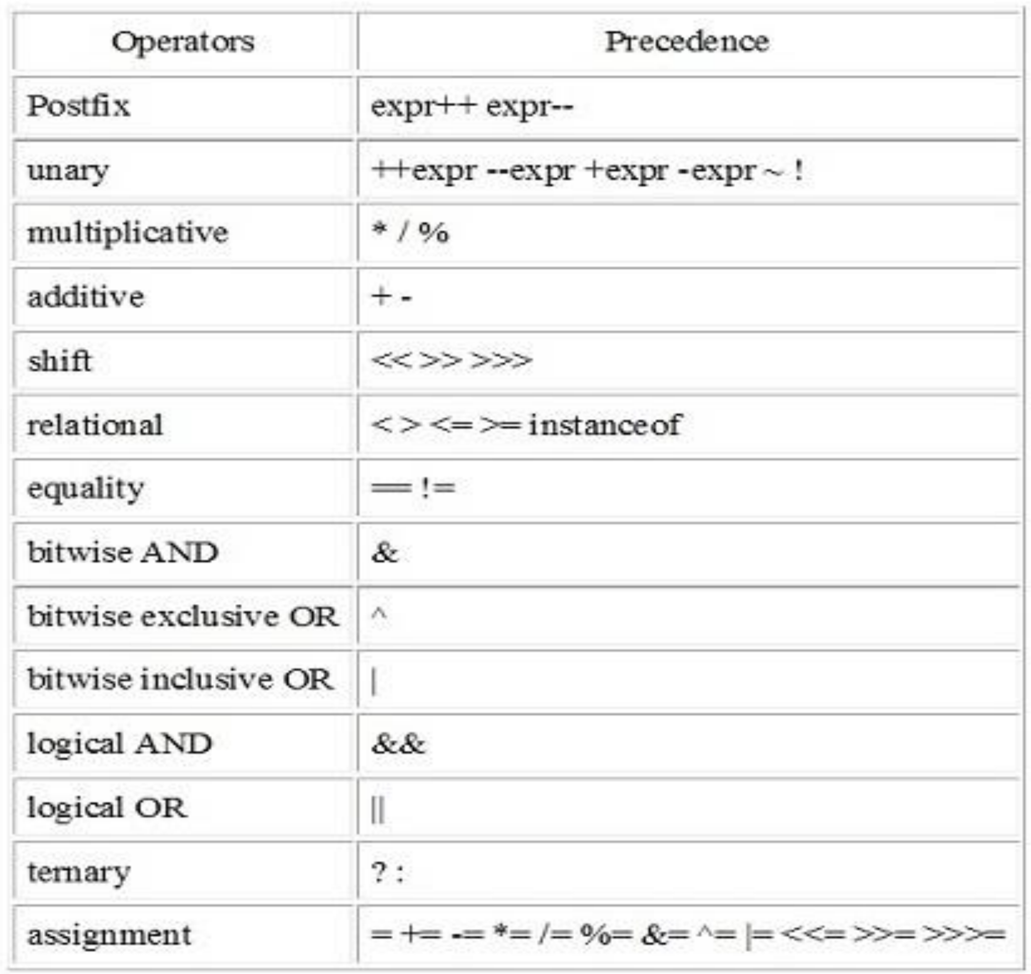
ContinueExample3.java

ContinueExample2.java

ContinueExample.java

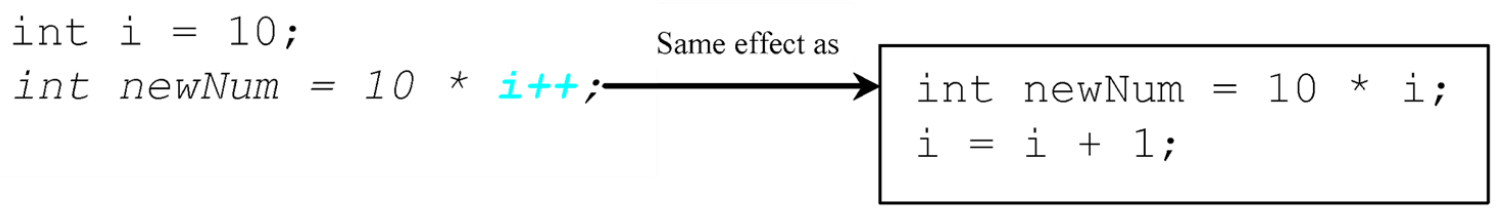
Operators in Java

Operators precedence in java



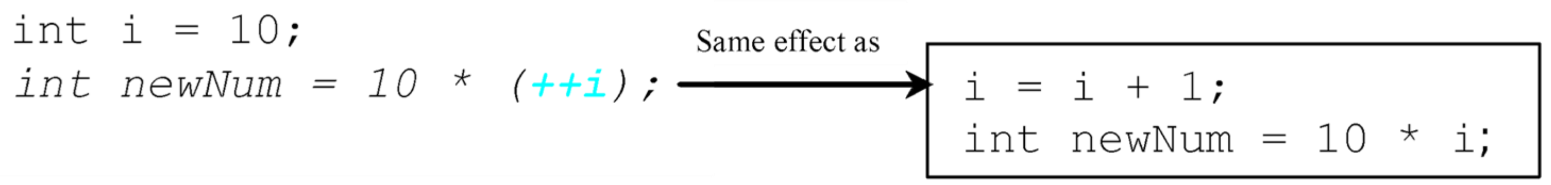
Operators	Precedence
Postfix	expr++ expr--
unary	++expr --expr +expr -expr ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Increment and Decrement Operators



```
int i = 10;
int newNum = 10 * i++;
```

```
int newNum = 10 * i;
i = i + 1;
```

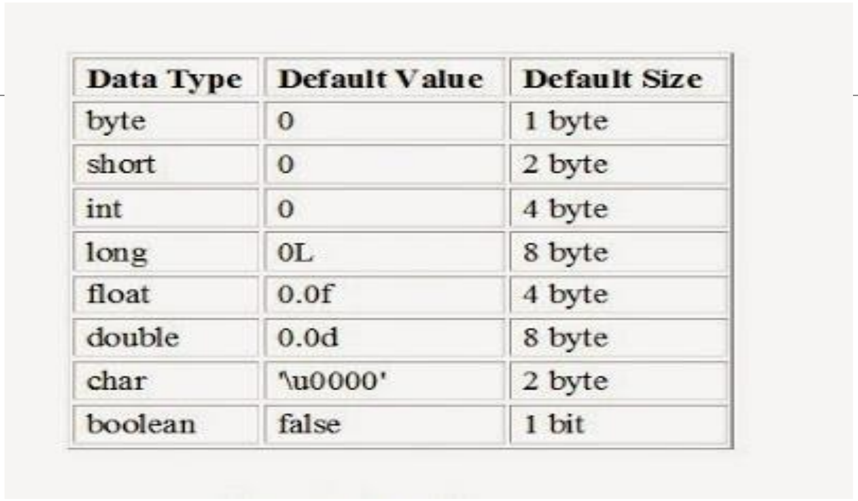


```
int i = 10;
int newNum = 10 * (++i);
```

```
i = i + 1;
int newNum = 10 * i;
```

Data Types in Java

- In java, data types are classified into two catagories :
- Primitive Data type



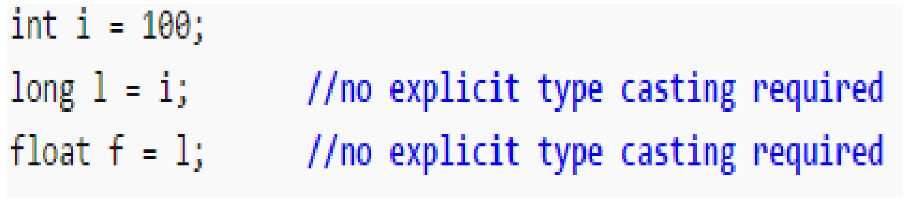
Data Type	Default Value	Default Size
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte
char	'\u0000'	2 byte
boolean	false	1 bit

- Non-Primitive Data type are Referenced or object data types:
  - is used to refer to an object. A reference variable is declare to be of specific and that type can never be change. Default value of object data types is null.

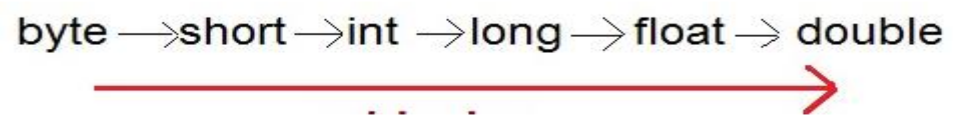
Note: Compiler never initialize local variable with default values. So you have to initialize local variable before using otherwise it will result in compile-time error

Type Casting

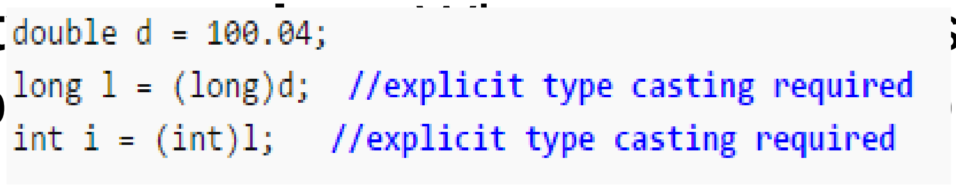
- Assigning a value of one type to a variable of another type is known as **Type Casting**.
- **Automatic Type casting (Implicit)** take place when, the two types are compatible and the target type is larger than the source type.



```
int i = 100;
long l = i;        //no explicit type casting required
float f = l;       //no explicit type casting required
```



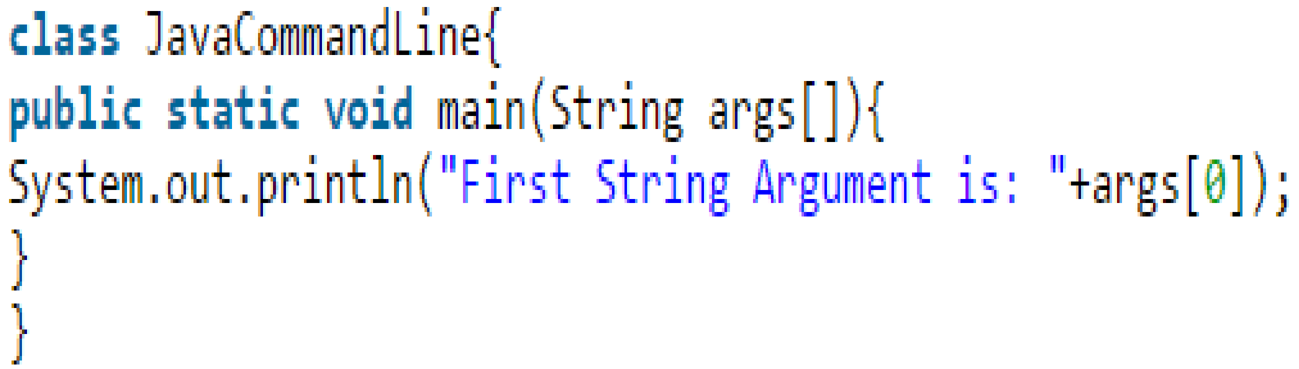
- **Explicit t** signing a larger type value to a variable o erform explicit type casting.



```
double d = 100.04;
long l = (long)d;  //explicit type casting required
int i = (int)l;   //explicit type casting required
```

Java Command Line Argument

- Command line arguments are the optional argument that supplied by the user or programmer at run time. In java command line arguments are handled by the main() function. It is an Array of string so that you can pass argument in the form of String only.



```
class JavaCommandLine{
public static void main(String args[]){
System.out.println("First String Argument is: "+args[0]);
}
}
```

Reading Input from the Console

1. import java.util.Scanner;

2. Create a Scanner object

```
Scanner input = new Scanner(System.in);
```

3. Use the methods next(), nextByte(), nextShort(), nextInt(), nextLong(), nextFloat(), nextDouble(), or nextBoolean() to obtain to a string, byte, short, int, long, float, double, or boolean value. For example,

```
System.out.print("Enter a double value: ");
Scanner input = new Scanner(System.in);
double d = input.nextDouble();
```

RAMAIAH
Institute of Technology

An Overview of Java

Object-Oriented Programming

- Object-oriented programming (OOP) is at the core of Java.
- In fact, all Java programs are to at least some extent object-oriented.

Two Paradigms

1. Process-Oriented Model
2. Object-Oriented Programming

An Overview of Java

Process-Oriented Model

- *This approach characterizes a program as a series of* linear steps (that is, code).
- The process-oriented model can be thought of as *code acting on data.*

Object-Oriented Programming

- This approach organizes a program around its data (that is, objects) and a set of well-defined interfaces to that data.
- An object-oriented program can be characterized as *data controlling access to code.*

OOP Principles

Abstraction is a process where you show only "**relevant**" data and "**hide**" unnecessary details of an object from the user.

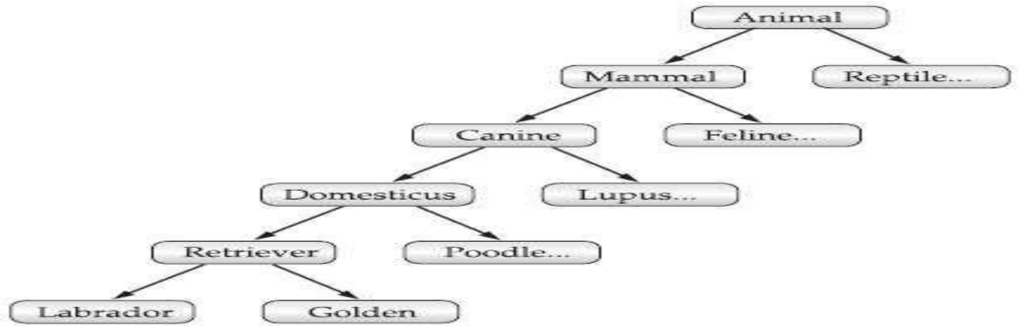
We use abstract classes and interfaces to achieve abstraction in java.

Ex: Working of a Car, Working of a Projector, Working of a Computer Working of an ATM.

OOP Principles

Inheritance

- When one object acquires all the properties and behaviors of parent object i.e. known as inheritance.
- It provides code reusability.
- It is used to achieve runtime polymorphism.



OOP Principles

Inheritance

- Without the use of hierarchies, each object would need to define all of its characteristics explicitly.
- However, by the use of inheritance, an object need only define those qualities that make it unique within its class.
- It can inherit its general attributes from its parent.
- Thus, it is the inheritance mechanism that makes it possible for one object to be a specific instance of a more general case.

OOP Principles

Polymorphism

- Polymorphism is a feature that allows one interface to be used for a general class of actions.
- In java, we use method overloading and method overriding to achieve polymorphism.
- Ex: Your mobile phone, one name but many forms: as phone, as camera, as mp3 player, as radio.

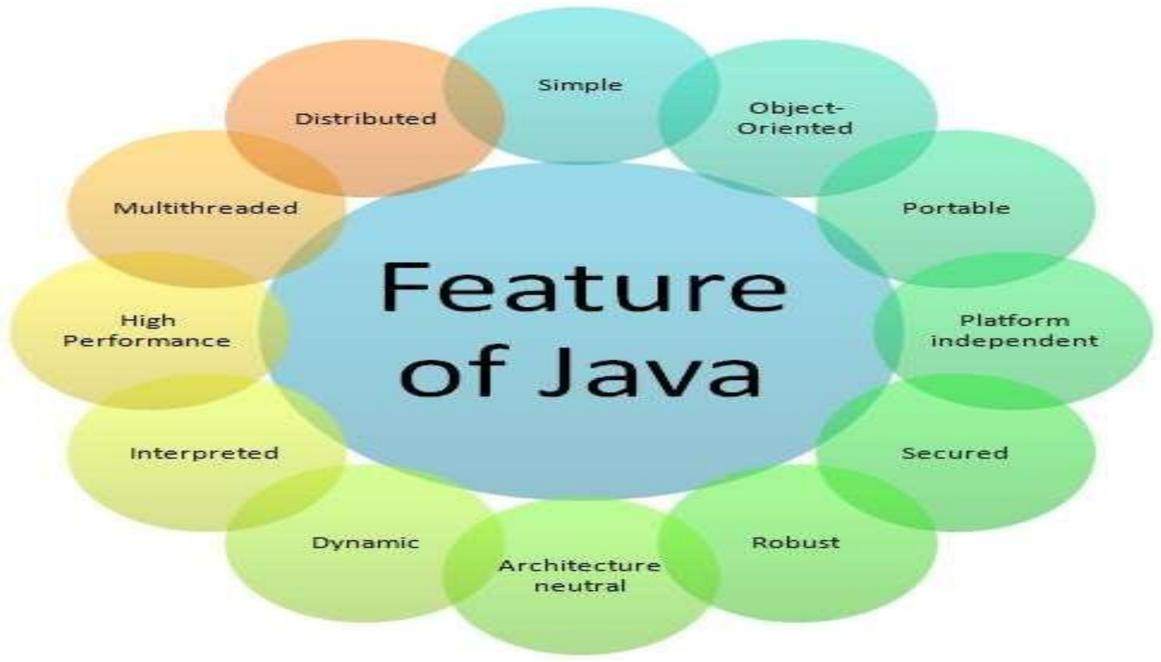
RAMAIAH
Institute of Technology

OOP Principles

Encapsulation

- Binding (or wrapping) code and data together into a single unit is known as encapsulation.
- Access to the code and data inside the wrapper is tightly controlled through a well-defined interface.
- A java class is the example of encapsulation.
- Ex: capsule, it is wrapped with different medicines.

Java Buzzwords



Java Buzzwords

Simple

- Java is a simple Language because it contains many features of other Languages like C and C++ and removes complexity because it doesn’t support pointers, storage classes, goto statements and multiple inheritances.

Object Oriented

- Java is purely an Object Oriented Programming language i.e., all the code of the Java language is written into the classes and objects.

Java Buzzwords

Distributed

- Java is a distributed language because, because of its ability to share the data and programs over the LAN.
- Access remote objects.

Multithreaded

- A Java program can be divided into multiple threads assigning different tasks for different threads and have all the threads executing in parallel.

RAMAIAH
Institute of Technology

Java Buzzwords

Dynamic

- The JVM maintains a lot of runtime information about the program and the objects in the program.
- Libraries are dynamically linked during runtime.

Architecture Neutral

- Java follows “Write-once-run-anywhere” approach.
- Java programs are compiled into byte code format which does not depend on any machine architecture but can be easily translated into a specific machine by a JVM for that machine.

Java Buzzwords

Portable

- In Java the size of the primitive data types are machine independent.
- Int in Java is always a 32-bit integer, and float is always a 32-bit IEEE 754 floating point number.

Interpreted & High Performance

- Java programs are compiled to portable intermediate form known as byte codes, rather than to native machine level instructions and JVM executes the byte codes on any machine on which it is installed.
- Just-in-time compiler.

Java Buzzwords

Robust

- A Program or an application is said to be robust (reliable) when it is able to give some response in any kind of context.
- Java’s features help to make the programs robust. Some of those features are: type checking, exception handling, etc.

Secured

- Java provides data security through encapsulation.

Java Buzzwords

Interpreted

- A Program or an application is said to be robust (reliable) when it is able to give some response in any kind of context.
- Java's features help to make the programs robust. Some of those features are: type checking, exception handling, etc.

Secured

- Java provides data security through encapsulation.

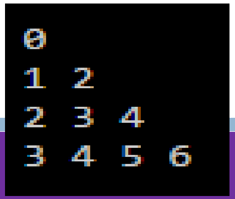
Exercises

1. Write a program that obtains hours and minutes from seconds using java operators
2. Write a program that converts a Fahrenheit degree to Celsius.
3. Write a java program that takes your first name and last name as command line arguments to the program and displays your name and last name on separate lines.
4. Write a program to print factorial of a number using loops.
5. Write a program that prompts the user to enter an integer from console. If the number is a multiple of 5, print HiFive. If the number is divisible by 2, print HiEven
6. Write a program print odd and even numbers in the following format using continue – break – label

```
0   1
2   3
4   5
6   7
8   9
```

Exercises on arrays

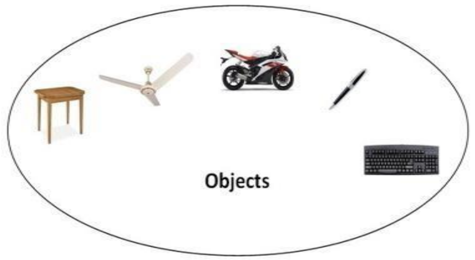
1. Set up an array to hold the following values, and in this order: 23, 6, 47, 35, 2, 14. Write a Java program to print out the highest number in the array.
2. Write a Java Program To Print Sum Of Upper Triangular Matrix
3. Write a Java Program to sort the numbers using bubble sort.
4. Write a Java program to print the numbers in the following format using Ragged array



```
0
1 2
2 3 4
3 4 5 6
```

RAMAIAH Institute of Technology

Object in Java



Object is the physical as well as logical entity

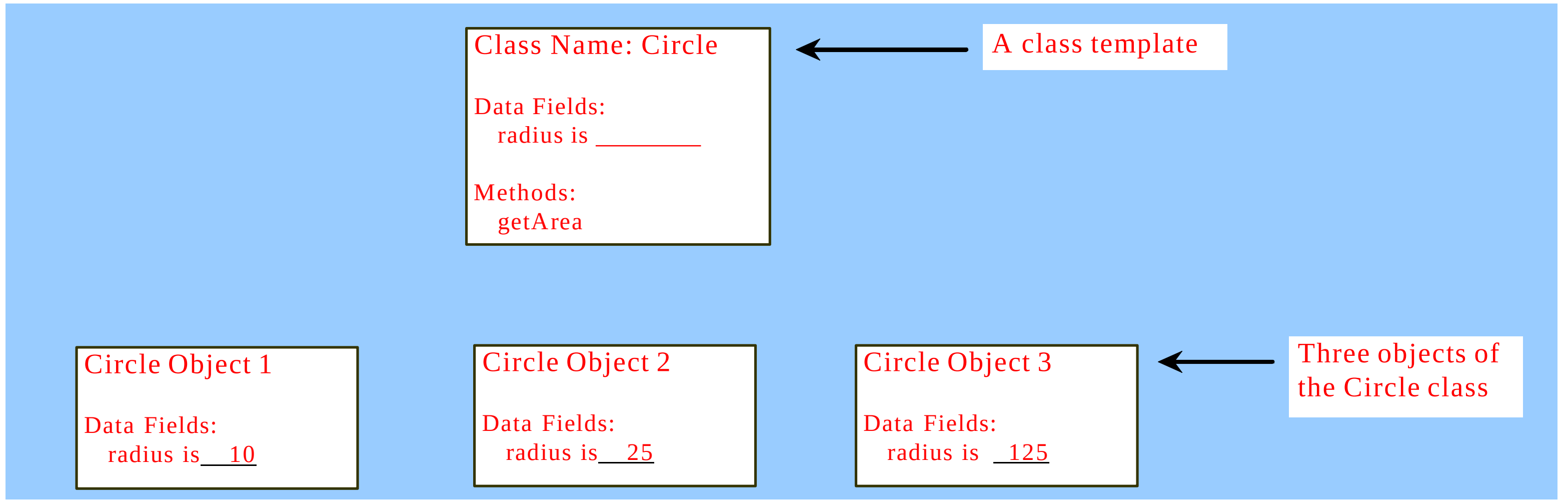
An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical (tengible and intengible). The example of integible object is banking system.

An object has three characteristics:

- **state:** represents data (value) of an object.
- **behavior:** represents the behavior (functionality) of an object such as deposit, withdraw etc.
- **identity:** Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.

For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state. It is used to write, so writing is its behavior.

Objects



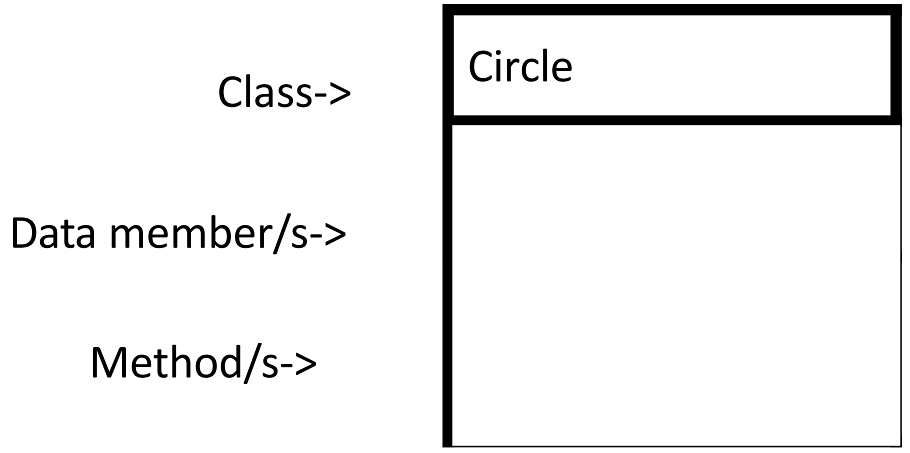
An object has both a state and behavior. The state defines the object, and the behavior defines what the object does

Class in Java

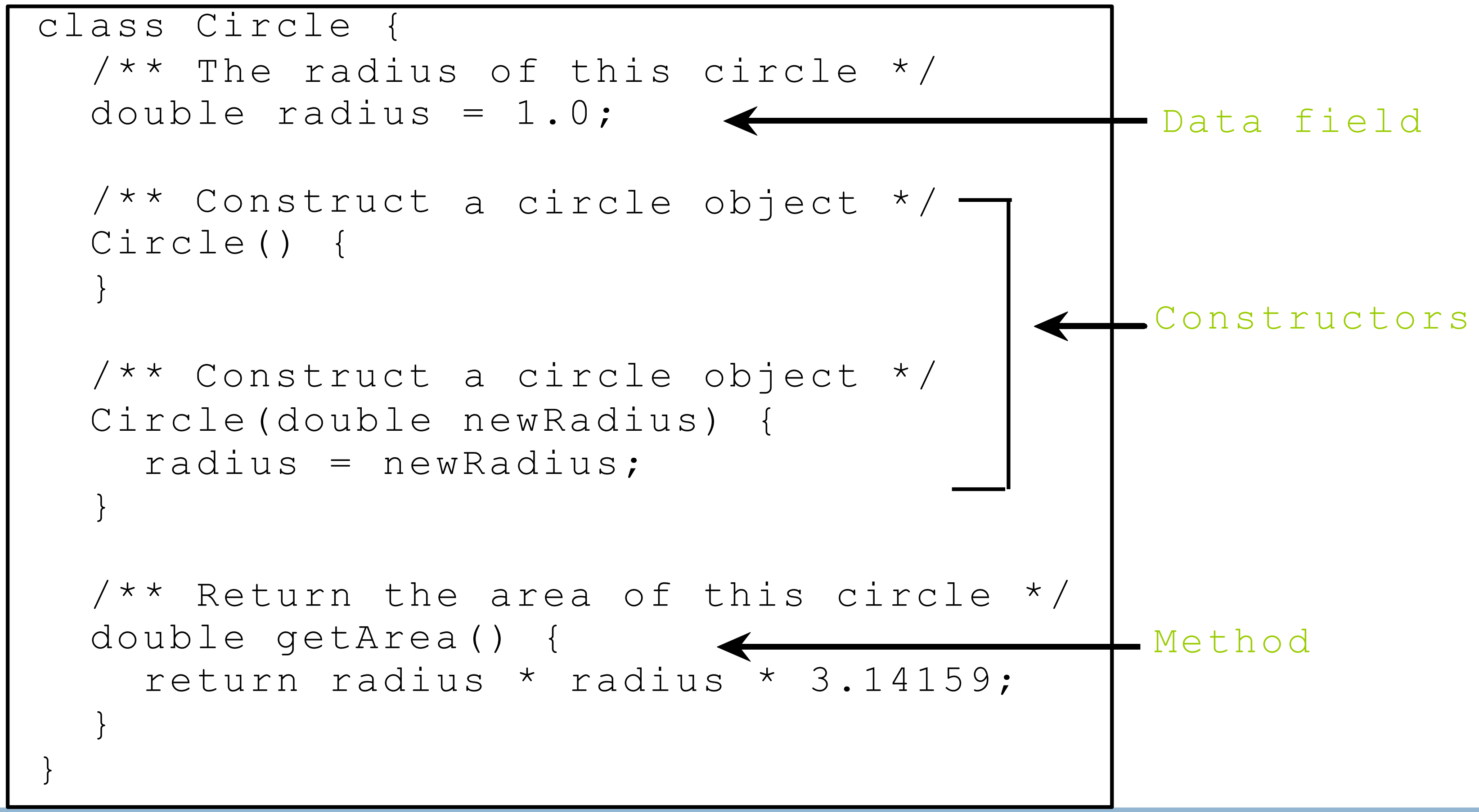
Object is an instance of a class. Class is a template or blueprint from which objects are created. So object is the instance(result) of a class.

A **class** is a group of objects that has common properties. It is a template or blueprint from which objects are created. A class in java can contain:

- **data member**
- **method**
- **constructor**
- **block**
- **class and interface**



Classes



```
class Circle {
  /** The radius of this circle */
  double radius = 1.0;

  /** Construct a circle object */
  Circle() {
  }

  /** Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }

  /** Return the area of this circle */
  double getArea() {
    return radius * radius * 3.14159;
  }
}
```

RAMAIAH Institute of Technology

Methods in Java

What is System.out.println? It is a method: a collection of statements that performs a sequence of operations to display a message on the console. It can be used even without fully understanding the details of how it works. It is used by invoking a statement with a string argument. The string argument is enclosed within parentheses. In this case, the argument is "Welcome to Java!" You can call the same println method with a different argument to print a different message.

Adding Methods to Class Circle

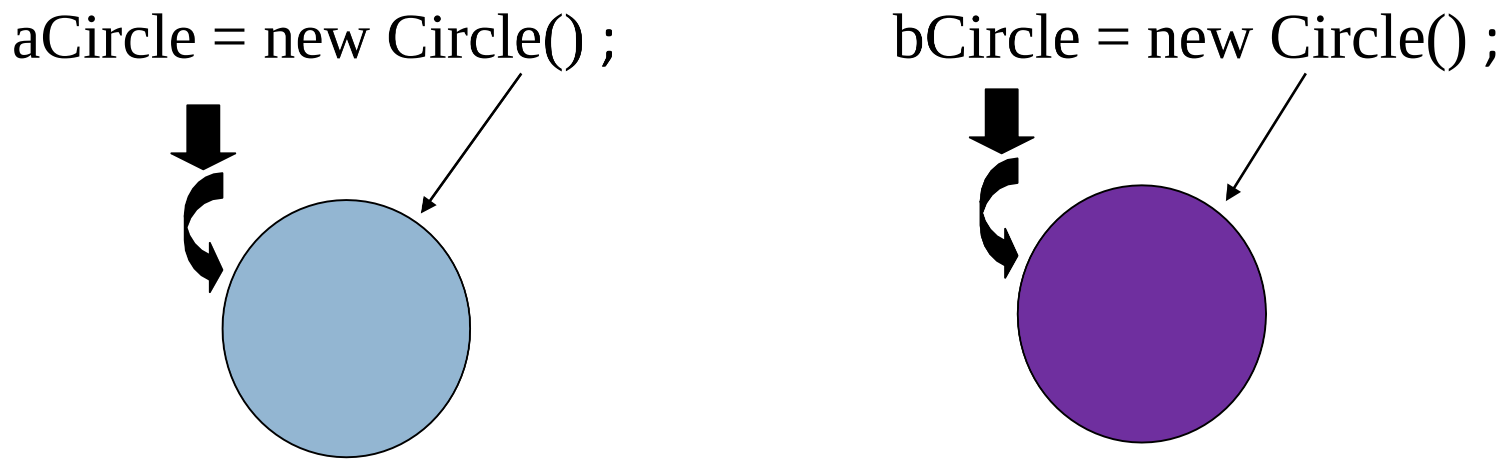
```
public class Circle {

    public double x, y; // centre of the circle
    public double r;    // radius of circle

    //Methods to return circumference and area
    public double circumference() {
            return 2*3.14*r;
    }
    public double area() {
            return 3.14 * r * r;
    }
}
```

Creating objects of a class

```
aCircle = new Circle();
bCircle = new Circle() ;
```

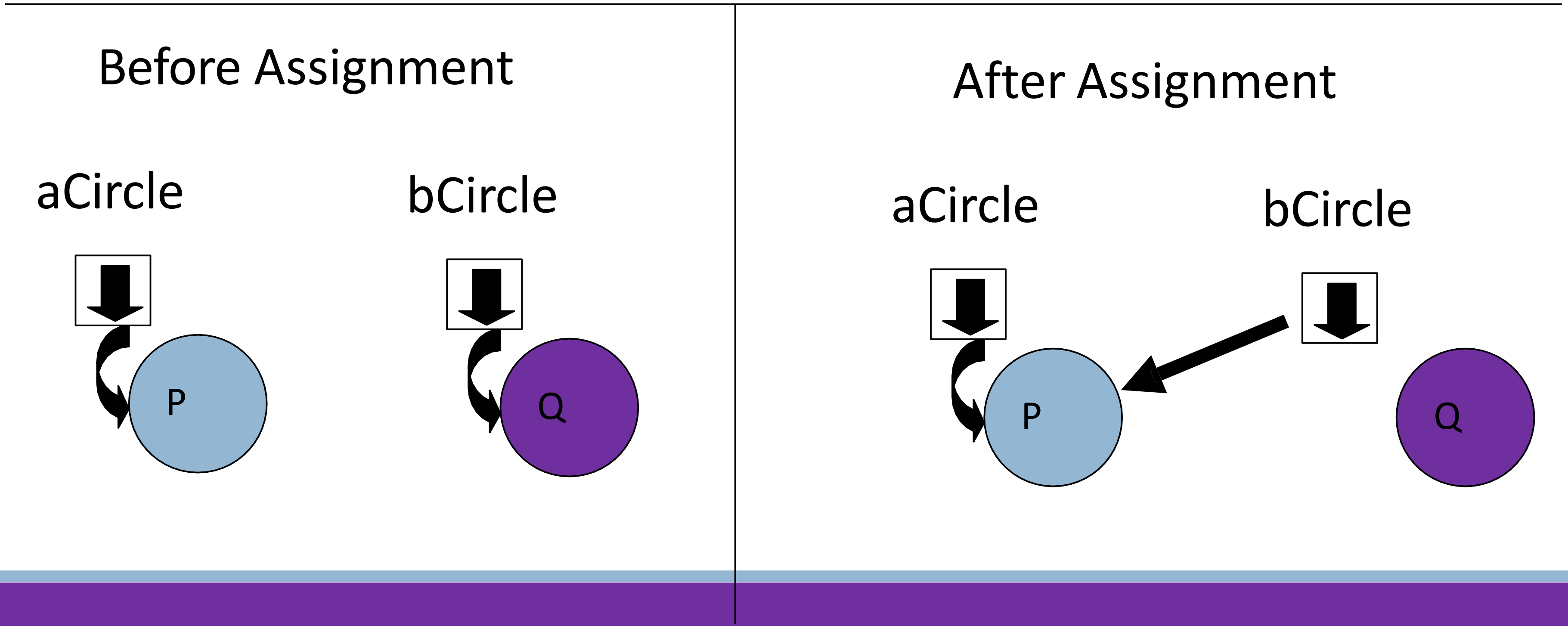
Objects are created dynamically using the *new* keyword.

aCircle and bCircle refer to Circle objects.



Creating objects of a class

```
aCircle = new Circle();
bCircle = new Circle() ;

bCircle = aCircle;
```



Accessing Object/Circle Data

- Similar to C syntax for accessing data defined in a structure.

```
Circle aCircle = new Circle();
aCircle.x = 2.0; // initialize center and radius
aCircle.y = 2.0;
aCircle.r = 1.0;
```

- Executing Methods in Object/Circle

```
double area;
aCircle.r = 1.0;
area = aCircle.area();
```

Using Circle Class

```
// Circle.java:  Contains both Circle class and its user class
//Add Circle class code here
class MyMain{
    public static void main(String args[])      {
        Circle aCircle; // creating reference
        aCircle = new Circle(); // creating object
        aCircle.x = 10; // assigning value to data field
        aCircle.y = 20;
        aCircle.r = 5;
        double area = aCircle.area(); // invoking method
        double circumf = aCircle.circumference();
        System.out.println("Radius="+aCircle.r+" Area="+area);
        System.out.println("Radius="+aCircle.r+" Circumference ="+circumf);
    }}
```

Thank you