Knowledge flow

JAVA FOR BEGINNERS

Knowledge flow

“LEARNING STARTS WITH VIEWING THE WORLD DIFFERENTLY.”

Knowledge flow- A mobile learning platform provides apps, eBooks and video tutorials

Knowledge flow brings you learning eBook of ***Java for Beginners***. This eBook is for all information technology and computer science students and professionals across the world.

Follow us on

Facebook

Google plus

Twitter

For more information visit us at

Knowledgeflow.in

knowledgeflowapps.blogspot.in

Thank you for using Knowledge flow eBooks

JAVA FOR BEGINNERS

Disclaimer

This eBook contents is for informational and study purposes only. The ***Knowledge flow*** makes no claims, promises, or guarantees about the accuracy, completeness, or adequacy of the contents of this eBook and no legal liability or other responsibility is accepted by ***Knowledge flow*** for any errors, omissions, or statements on this eBook.

Introduction to Java

- Java was designed and conceived by ***James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sherdan***, which was done at Sun Microsystems in year 1991.
- It took almost 18 months for java to come into existence as a working version.
- Initially java was known as “Oak”, which was then renamed as “Java” in year 1995. Since java had much had much of its character designed from C and C++.
- This character inherited by the two well known and simple programming makes java more appealing to computer and it giants which would lead to a large scale success.
- But java is misunderstood as the sophisticated internet version representation of C++.
- It has significant difference practically and philosophically when compared to C++.
- If you have good knowledge in C++ then you will find java as your cup of tea and you will at ease using and understanding java.

Logo of Java

Therefore, there are two main reasons for the evolution of the computer languages.

Java had enhanced and refined the object oriented scenario of C++. This gave more features to the users which are as follows.

- Multithreading.
- Library which would provide easy internet access.
- One of the java’s magic was the byte code. Byte code is set of instruction which is

highly optimized and designed to be executed by JVM (Java Virtual Machine). It is an interpreter for byte code. This lead to the design of truly portable programs.

Java redesigned the internet with new feature and networked program which are as follows.

- *Applets* - It is a kind of java program that is to be transmitted over and executed automatically by java compatible web browsers.
- *Security* - It provided the security of downloading various applets and programs from internet without containing any virus or Trojan horses.
- *Portability*- Since there is large and different kind of operating systems therefore it provides the freedom of running in any operating system so its program can be used in different OS without any issues of compatibility.

The evolutions in java are as follows.

- Java 1.0
- Java 1.1
- Java 1.2
- J2SE
- J2SE 1.2
- J2SE 1.3
- J2SE 1.4
- J2SE 5
- J2SE 5 made various changes to Java

The new feature that was added is as follow.

- Generics
- Annotations
- Auto boxing and auto-unboxing
- Enumerations
- Enhanced, for-each style for loop
- Variable-length arguments
- Static import
- Formatted I/O

- Concurrency utilities

In J2SE 5, and the developers kit was called JDK 5. 1.5 used as internal version number and this is referred as developer version number.

Java became the center of innovation in computer technological world. The existence of JVM and byte code changed the scenario of security and portability in the programming world. The way the new ideas are put into the language has been redefined by the JCP i.e. java community process.

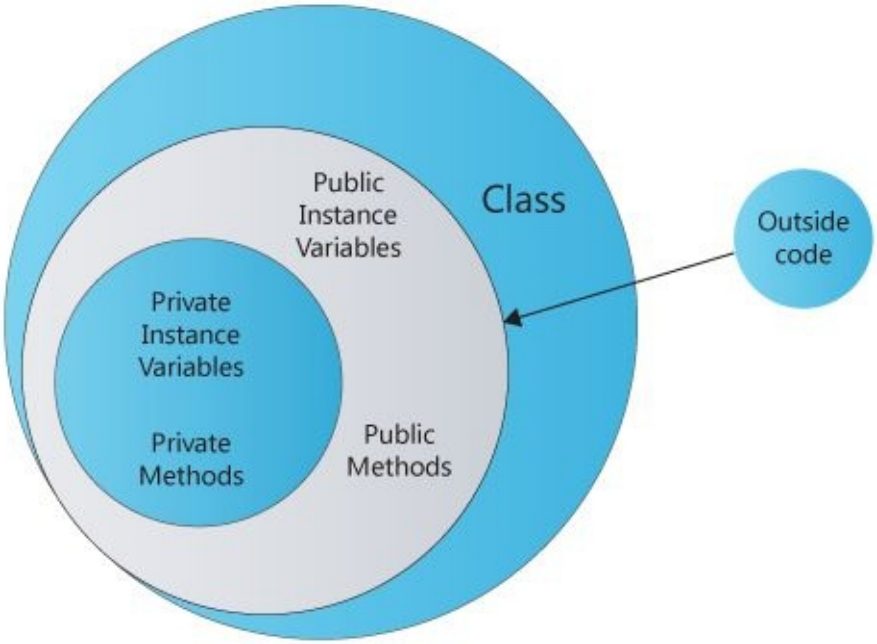
Features of Java

Object-oriented programming

- This is the core feature of java.
- This is to manage the increase in the complexity.
- It provides a very sophisticated and well defined interface for the data.
- It is also known as data controlling access code.
- Another important feature of java being object oriented is abstraction.
- Complexity can be managed using abstraction.

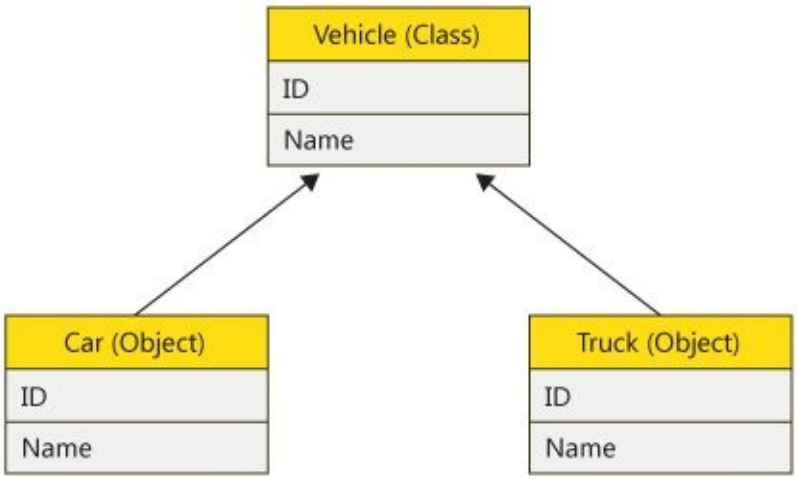
The three OOP principles

- *Encapsulation*- Its agenda is to manipulate the data and keep the data isolated and safe from the external interference and misuse. The encapsulation is done by the use of the protective wrapper. This prevents the external sources from accessing the data or the code.



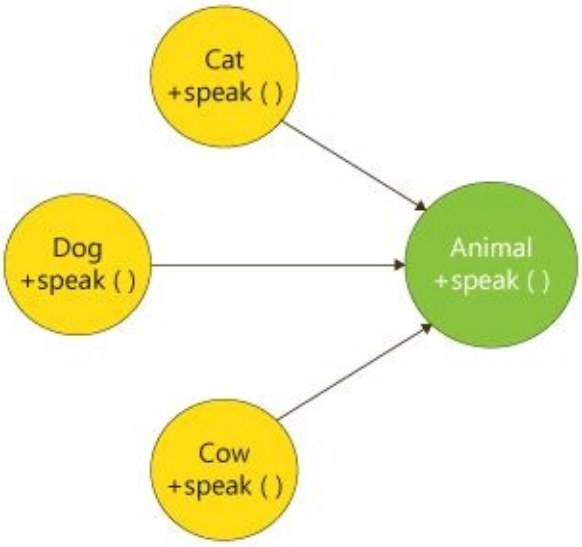
Encapsulation in Java

- *Inheritance*- In this the object would acquire the property of other object present. It just follows the concept of the hierarchical classification. This consists of classes, sub classes. Inheritance also is linked or interacts with encapsulation as well.



Inheritance

- *Polymorphism*- It is means many ways to carry out the method but from one input.



Polymorphism

Byte code

This is highly optimized by set of instructions designed which is designed to be executed by Java virtual machine that is JVM.

JVM

- It was designed as an interpreter for the byte code.
- Another feature of java program is that it is simple.
- This enables the professionals to learn.
- Work in a very effective manner but it is also very easy to understand.

Robust

The ability that includes creating a robust program that can be a multiplatform program are given a very high priority in design of Java.

Multithreading

The real world requirements are met by java which helps to achieve the requirement of creating interactive and networked programs.

High performance

- The advantage of being a multi platform functioning program helps to find the cross platform solution.
- It provides benefits of being an platform independent code with the help of java run time system.

Distributed

- This is because it is been designed for the internet which has a distributed environment because of the handling of TCP/IP protocols.
- This allows the program to find out methods across a network.
- URL is used in this to access a file on internet.
- This property supports RMI (Remote Method Invocation).

Dynamic

- This is the action that is taken during the run-time such as to resolve, verify and add objects.
- It provides us the function which will allow us to link code dynamically that will be safe.

Simple program

```
/* Call this file “Example.java” */

Class Example {

Public static void main (string args []) {

System.out.println (“this is a simple java program.”);

}

}
```

Command line argument to pass the class name is

```
C :\> java Example
```

Simple output of the above program

this is a simple java program.

Calling of the file in java cmd

Calling of the file: “Example.java”.

Data Types

Integers

It includes whole valued signed numbers which are as follow.

Byte- Whose width is 8 and it ranges from -128 to 127. It is the smallest integer type. It is useful when user is dealing with stream or flowing data from a network or a file.

Declaration of two byte variables called x and y is as follow

```
Byte x, y;
```

Short- The width of this type of integer is 16 and it ranges from -32,768 to 32,767. It is the least commonly used data type in java.

Declaration

```
Short a;

Short b;
```

Int- The width of this type of integer is 32 and it ranges from -2,147,483,648 to 2,147,483,647. It is more efficient as compared to byte and short. It is commonly used to control loops and indexed arrays.

Declaration

```
Int a;
```

Long- The width of this type of integer is 64 and it ranges from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. This is used for those values which are large enough that the integer cannot handle them.

Declaration

```
Int light speed;

Long days;

Long second;

Seconds = days *24 *60*60;
```

Floating point type

This is for real numbers used for calculation such as square roots, sine and cosine. These are of two types.

Float- Width in bits is 32 and range is from 1.4e-045 to 3.4e + 038. Float is used as a variable type for the fractional component, but specifies single precision. It can be used in representing dollars and cents.

Declaration

```
Float hightemp;
```

Double- Width in bits is 64 and range is from 4.9e-324 to 1.8e + 308.it is used and is optimized in such a way that I can be used for high speed mathematic calculation. It has double precision which is faster than the single precision. Function such as sin()and sqrt() etc, return double values.

Declaration

```
Double pi, r, a;

r = 10.4;

pi = 3.14;

a = pi* r * r;
```

Characters

- The data type used store or declare character in java is char.
- It is 16-bit type in java. Range of a char is 0 to 65,536.
- There are no chars which are negative in nature.

Program demonstration for char

```
Class charExample

{

Public static void main (String args []) {

Char a, b;

a=88;

b=’y’;

system.out.print (“a and b: “);

system.out.println (a + “ “ +b);

}
```

}

Booleans

- It is a primitive type for logical values.
- This tends have only one of two possible values, true or false.
- It is governed by if or for control statements.

Variables

It is basic unit of storage and variables have scope, visibility and lifetime.

Declaration of variables

Int x, y, z;

Or

Int x=1, y=2, z=5;

Type conversion and casting

If there is any compatibility existing between two types then java would automatically performs the conversion.

Java automatic conversion

- It will be performed when two conditions are met.
- When the two types are compatible;
- Destination type is larger than the source type.
- After the two conditions are been satisfied a widening conversion takes place.

Casting compatible types

- It is a simple and explicit type of conversion.
- When a floating type of conversion is attached to an integer type this type of conversion is called truncation.
- It is a kind of conversion sometime called narrowing conversion.

Arrays

- It is a group of similar variables that would be referred by a common name.
- The element available in the array is accessed through index.
- This can be created and may have one or more dimension.

Types of arrays

There are two main types of arrays are.

1. One-dimensional arrays
2. Multi-dimensional arrays

One-dimensional Arrays

It's a list of similar types of data. Before you create array you need to create a variable of any type.

Syntax

Type variable-name [ ];

Int year_month [ ];

Multi-dimensional Arrays

It is arrays of arrays need to specify each dimensional array variables, additional index using other square brackets represented as.

Int twoD [ ] [ ] = new int [6] [7];

Operators

There are four main types of operator.

- Arithmetic operators
- The Bitwise operators
- Relational operators
- Boolean logical operators

Arithmetic operators

- *Addition*- This operator is used to add the values. It is represented by "+". *For example* A = a + 4.
- *Subtraction*-This operator is used to subtract the values. It is represented by "- ". *For example* A = a - 4.
- *Multiplication*- This operator is used to multiply the values. It is represented by "*". *For example* A = a * 4.
- *Division*-This operator is used to divide the values. It is represented by "/". *For example* A = a / 4.
- *Modulus*- This operator is used to find the remainder of the values when divided. It is represented by "%". *For example* A = 2 % 4. Where A = 0.
- *Increment*-This operator is used to increases its operand by one. It is represented by "++". *For example* a = a++ which is equal to a= a+1.
- *Decrement:* This operator is used to decreases its operands by one. It is represented by "—". *For example* a = a— which is equal to a= a-1.

Bitwise operators

Unary NOT- This inverts all of the bits of operand contained and it is represented by "~".

For example

~00101010 = 11010101

And- It produces 1 bit if both operands are also 1 and it is represented by "&".

For example

00101010

&00001111

000101010

OR- If either of the operand is one it produces 1 and It is represented by "|".

For example

00101010

|00001111

00101111

XOR- If either of the bit operand is 1, then result is also one otherwise its 0 and it is represented by “^”.

For example

00101010

^00001111

00100101

Left shift- It shifts or moves all of the bits in the particular given value to the left side number of times that is been declared and it is represented by ” ≪ ”.

Right shift- It shifts or moves all of the bits in the particular given value to the right side number of times that is been declared and it is represented by ” ≫ ”.

Relational operators

Equal to- This relation operator shows that the values are equal to each other and it is represented by “==”.

Not equal to- This relation operator shows that the values are not equal to each other and it is represented by “!=”.

Greater than- This relation operator shows that one value is greater when compared to other and it is represented by “>”.

Less than-This relation operator shows that one value is less when compared to other and it is represented by “<”.

Greater than or equal to- This relation operator shows that one value is greater or equal but not less when compared to other and it is represented by “>=”.

Less than or equal to-This relation operator shows that one value is smaller or equal but not greater when compared to other and it is represented by “<=”.

Boolean logical operators

Logical AND- It is represented by “&”.

For example

A & B = If a is false and b is true it results as false, but is both are true it results as true. Similarly when both are false it results as false.

Logical OR- It is signed |.

For example

A | B = If a is false and b is true it results as true, but is both are true it results as true. Similarly when both are false it results as false.

Logical XOR- It is shown using "^".

For example

A ^ B = If a is false and b is true it results as true, but is both are true it results as false. Similarly when both are false it results as false.

Logical Unary Not- It is depicted as "!".

For example

! A = If a is false it results as true, but if a is true it results as false.

Assignment operators

The assignment is the single equal sign that is represented by "=".

The general representation

var = expression;

In this the variable, which is represented as var should be compatible with the type of expression.

Int a, b, c;

a = b = c = 100;

The ? Operator

This is used for replacement of if then else statements and it is represented as "?".

For example

expression a ? expression b : expression c

This above example states that if the expression *a* is true then expression b is evaluated otherwise expression c is evaluated.

Control Statements

There three types of statements in java.

1. Selection statements
2. Iteration statements
3. Jump statements

Selection statements

- It is to manage the flow of programs that is to be executed based on the dynamic conditions which can be only realized during the run time.
- It provides flexibility.

If statement- It provides different paths for execution of program.

Syntax

If (condition provided) statement a;

Else

Statement b;

This means if the condition is true, then statement *a* is executed but if false then statement *b* is executed.

For example

int a, b;

If (x < y) a=0;

Else

b=0;

Now there is one type of procedure using if statement *i.e.* nested if statement and it is very common method in programming world.

For example

If (a == 10) {

If (b < 15) i = j;

If (c > 50) p = q;

Else

i = p;

}

Else

i = q;

Now, the second if statement in the parenthesis is associated with else. Another type of procedure is of using if statement is the if-else-if ladder statement.

For example

If (condition)

Statement;

Else if (condition)

Statement;

Else if (condition)

Statement;

.

.

.

.

.

Else

Statement;

Switch statements

It is a multi branched statement.

Syntax

Switch (expression)

{

Case value 1:

Statement

Break;

Case value 2:

Statement

Break;

.

.

.

Case vale n:

Statement

Break;

Default:

Statement;

}

The functioning of the switch statement is compares the given value with all the cases and when the match found the program sequence is executed.

For example

Class sampswitch {

Public static void main (string args [])

{

Switch (i)

{

Case0:

System.out.println (“i is zero.”);

Break;

Case1:

System.out.println(“i is one.”)

Break;

Default:

System.out.println (“i is greater than 4.”)

}

Output

i is zero.

i is one.

i is greater than 4.

i is greater than 4.

Example of nested switch loop program

Switch (countfigures)

{

Case 1:

Switch (targetvalue)

{//nested switch//

Case 0:

System.out.println (“target is zero”;

Break;

Case 1:

System.out.println (“target is one”;

Break;

}

Break;

Case 2: //…

Important features of switch statement are.

- Switch statement can only check for equality unlike if.
- There cannot be two cases constant in the same switch with identical values.
- It is more effective than using the set of nested if statement.

Iteration statements

These statements are used to create loop. There are three types of iteration statement used in java.

While statement

Its function is to repeat or block statement while its controlling expression is true.

Syntax

```
While (condition) {
// body of loop
}
```

The loop is executed when the condition is true. When it is false the control will pass to the next line of the code following the loop.

For example

```
Class while {
Public static void main (string args [])
{
Int n = 5;
While (n>0)
{
System.out.println (“tick “+ n);
n—;
}
}
}
```

Output

It will tick 5 times

Tick 5

Tick 4

Tick 3

Tick 2

Tick 1

Do-while statement

This loop executes its body at least once because it contains its condition expression at the bottom of the loop.

Syntax

do

{

System.out.println (“tick”+ n);

n— ;

}

While n > 0);

}

}

For statement

- It is a very powerful as well as very versatile construct.
- The condition what is provided in for statement until and unless the condition is not been satisfied it won’t execute.
- If the condition is evaluated and the value is false then it is terminated.

Syntax

For (initialize; condition; iterate)

{

Body

}

For example

Class fortick

{

```
Public static void main (string args [])
{
Int n;
For (n=11; n>0; n—);
}
}
```

Jump statements

Java supports three jump statements which are as follows.

1. Break
2. Continue
3. Return

Break

It is used to terminate a statement which is in sequence in a switch statement and it can be used for exiting a loop. Some time It can be used instead of goto.

Continue

It provides an early iteration for a loop.

For example

```
Class continue
{
Public static void main (string args [])
{
For (int i=0; i<5; i ++)
{
System.out.print (i + ““ );
If (i%2 ==0) continue;
System.out.print (““ );
}
```

}

}

Return

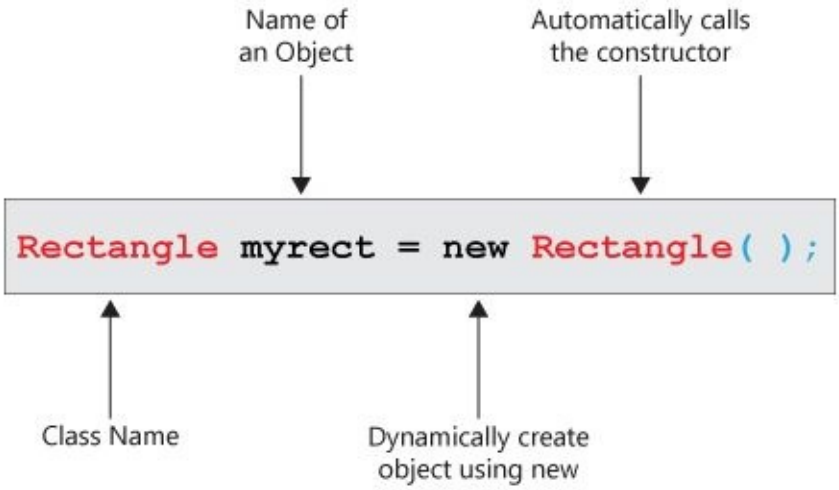
It is used to return from a method and It is a transfer back calling of the method.

For example

```
Class return {

Public static void main (string args [])

{

Boolean t = true;

System.out.println (“return”);

If (t) return;

System.out.println (“this won’t work”);

}

}
```

Classes and Methods

Classes might contain only code or data but may also contain both. A class is declared by keyword *class* and it can be complex in nature.



Syntax

Class classname

{

Type instance-variable a;

Type instance-variable b;

.

.

.

Type instance-variable n;

}

Type method a (parameter-list)

{

//body of method

}

Type methodname a (parameter-list)

//body of method

.

.

}

}

Classes are also called the box which defines variable- width, height and depth.

Example

```
Class box

{

Double width;

Double height;

Double depth;

}
```

Class that implements stack of data and integer.

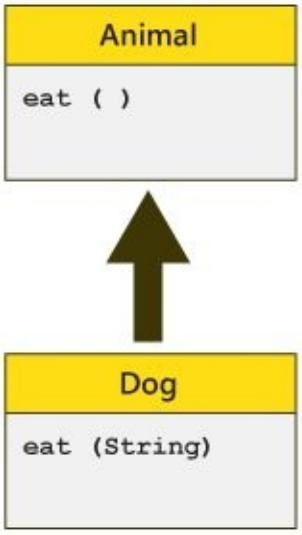
Example

```
Class stack

{

Int stk [] = new int [10];

Stack ()

Tos=-1

}

Void push (int item) {

If (tos==9)

System.out.println (‘it’s full”);

Else

Stk [++tos] = item;
```

```
}

Int pop ()

{if (tos<0) {

System.out.println (“stack underflow”);

Return 0;

} else

Return stk[tos—];

}

}
```

Overloading method

When two or more methods are defined in the same place and the name defined is same but the parameters are different. This method is an overloading method. This provides polymorphism in java.



Example

```
Class overload1

{

Void test () {

System.out.println (“no parameter”);

}

Void test (int a)

{

System.out.println (“a:  “+ a);

}

Void test (int a, int b)

{
```

```
System.out.println (“a and b: “+ a + ““+ b);

}

Double test (double a)

{

System.out.println (“double a: “+ a);

Return a*a;

}

}

Class overload

{

Public static void main (string args [ ])

{

Overload1 ob = new overload ();

Double result;

ob.test ();

ob.test (10);

ob.test (10, 20);

Result = ob.test (123.25);

System.out.println (“result:” +result);

}

}
```

Output

A= 10

A and B= 10 20

Double a= 123.25

Result= 15190.5625

Inheritance

Inheritance is a super class from which all the classes are inherited and which does inheriting from a super class is called the sub class.

Basic program of inheritance

```
Class x

{

int i, j;

void show I j ( )

{

System.out.println (“i and j:” + i + ““+ j );

}

}

Class y extends x

Int a; {

System.out.println (“a: “+ a );

}

Void sum ( )

{

System.out.println (“i + j + a: “+ (i + j + a) );

}

}

Class simpinheritance

{

Public static void main (string args [ ])

{

q Superob = new q ( );
```

```
p superob = new p ( );
Subob. i = 8;
Subob. j = 7;
Subob. a = 9;
System.out.println (“contents”);
Subob.showij ( );
Subob.showa ( );
System.out.println (“sum of i, j and a”);
}
}
```

Output

Contents: I and j: 8 7

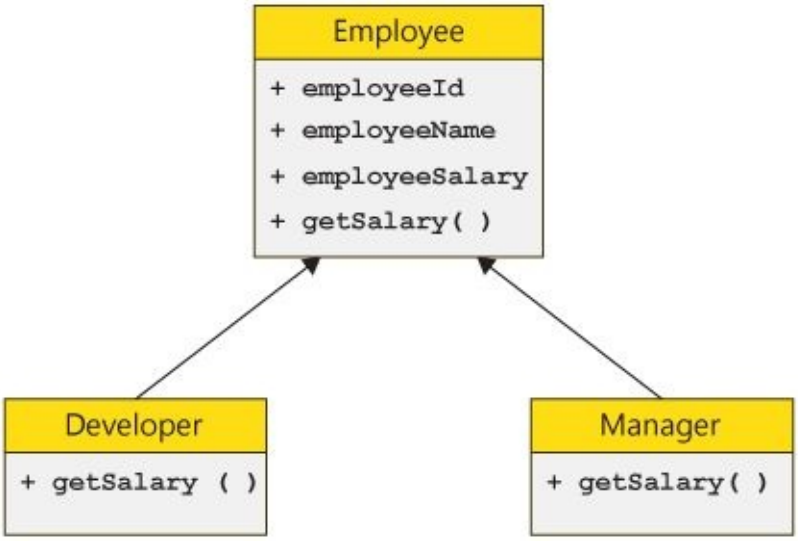
a: 9

Sum of i, j and a: i + j +a: 24

Main use of inheritance in java is for overriding and code reusability.

Overriding

- When sub classes are identical of a super class this scenario is called method overriding.
- It provides the characteristic called polymorphism in java.



Inheritance showing overriding

Example code

```
Class x {
int a, b
X (int p, int q)
{
p= a;
q= b;
}
Void show ( )
{
System.out.println (“a and b: “+ i +” “+ j);
}
}
Class y extends x {
Int r;
Y (int a, int b, int c)
{
Super (a, b);
r = c;
}
Void show ( ) {
System.out.println (“r: “+ r );
}
}
Class override {
Public static void main (string args [])
{
```

```
y subob =new y (1, 2, 3);
Subob. Show ();
}
}
```

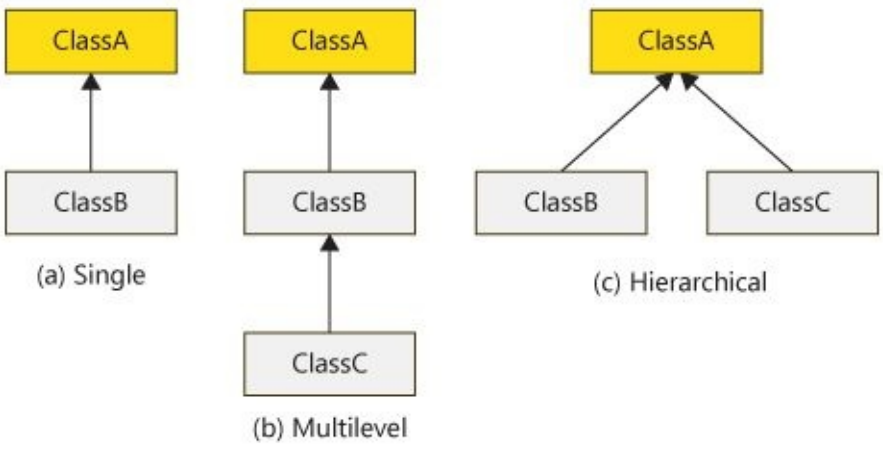
Output

```
r = 3
```

Types of inheritance

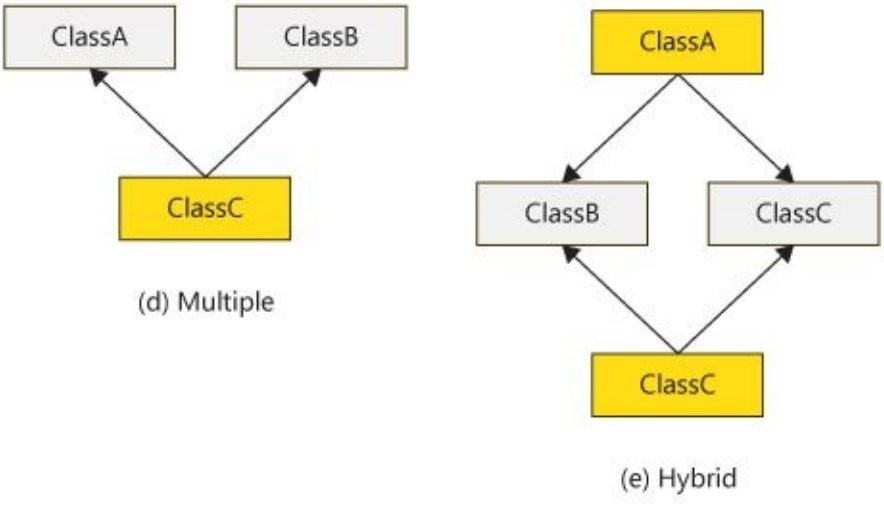
There are three types of inheritance.

- Single inheritance
- Multiple inheritance
- Hierarchical inheritance



Types of inheritance

In java multiple and hybrid will only supported with use of interface.



Multiple and hybrid inheritance

Packages

Packages contain names were classes are stored.

Syntax

Package pkg;

To represent multi packaging

Package pkgx [.pkgy [.pkgz]];

For example

Package java.awt.image a;

Finding of packages

There are three ways to find the packages. The java runtime system uses the working directory; if the package is contained in the sub directory then it will be discovered or you can set the location by setting the location of class as shown.

Package mpack

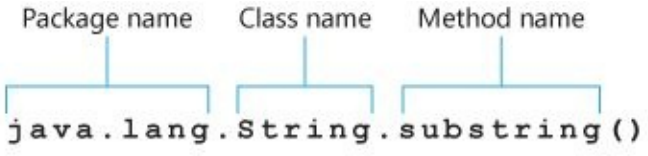
It can use class path with java as well as javac to specify the location of class as shown.

C: \ myprograms\java\mpack

Access protection

For the protection of packages there are three specifiers in existence in java.

1. Private- cannot be seen or accessed from outside the class.
2. Public- it has feature to be accessed from anywhere.
3. Protected- this can be accessed only by classes which are sub-classes.



Package syntax

Importing a package

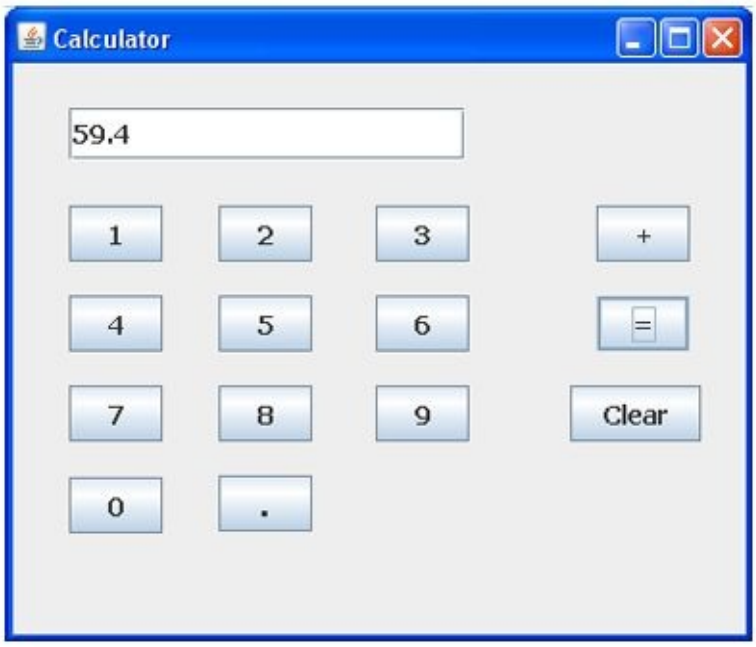
The statement that imports the package come immediately after the packaging statement and its main function is to import the package by saving more time to its location.

Syntax

Import pkga [.pkgb]. (classname d *);

Interface

It is similar to class which contains constants, methods and signatures.



Java simple calculator interface

Interface for java

Declaration

Access interface m

{

Return method 1 (parameter list);

Return method 2 (parameter list);

Type final variable name 1 = value;

Type final variable name 2 = value;

//…

Return type method name n (parameter list);

Type final varname n = value;

}

Simple interface contains only one function or method known as callback function.

For example

```
Interface callback
{
Void callback (int parm);
}
```

Implementing interface

To creating any interface implementation of class is included in definition of a class.

For example

```
Class client call back
{
Public void callback (int x)
{
System.out.println (“callback” + x);
}
}
```

Exceptional Handling

It is a drawback that affects the performance of java program because it can occur at any point. It occurrence is at execution state.

Reasons

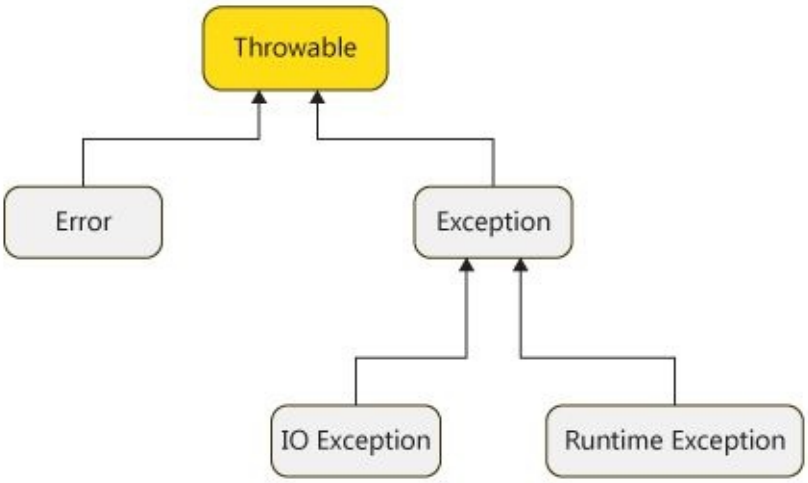
- Invalid data
- File absent
- Loss of network connection and time out when the JVM is at run.

The common exceptions are.

Checked- These exceptions cannot be forecasted by the programmer and it cannot be even ignored during.

Runtime- This is a dynamic exception, but it can be ignored unlike the checked one.

Errors- These are also ignored because nothing can be done to overcome it and this is not under programmer's control.



Exception hierarchy

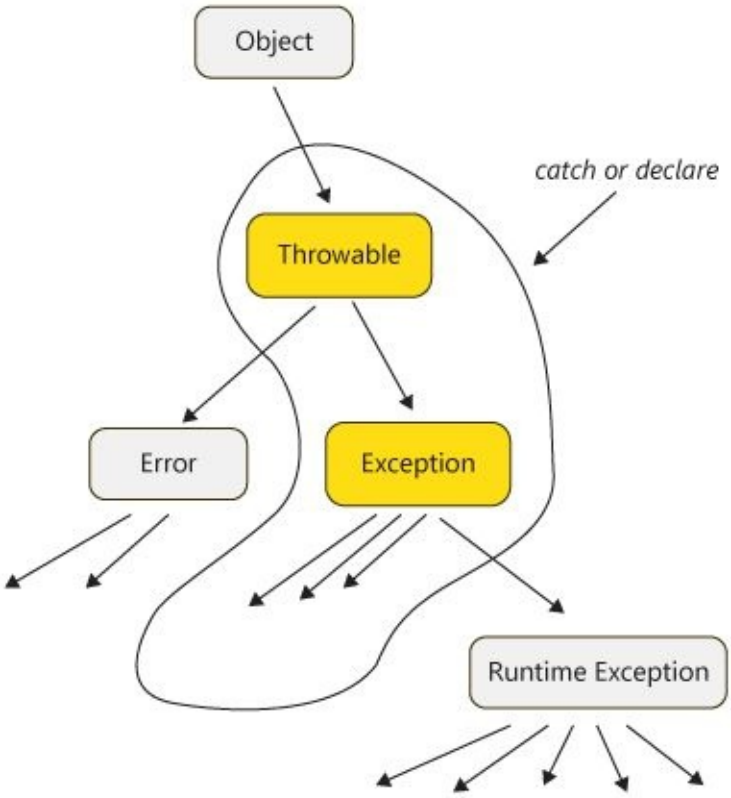
It has two sub divisions: Input output exception classes and runtime exception class.

Throw

These are exceptions which are thrown by JVM during runtime.

Syntax

throw Throwableinstance;



Throw

When this statement is used there is an immediate stop in the execution of program.

For example

```
Class throwdem
{
Static void demopro ( )
{
Try {
Throw new NullPointerException (“dem “);
} catch (NullPointerException e)
{
System.out.println (“caught inside demopro. “);
Throw e;
}
}
Public static void main (string args [ ] {
```

```
Try {
Demopro ( );
}
Catch (NullPointerException e) {
System.out.println (recatch: “+ e);
}
}
}
```

Finally

It is a code that is used in executing before or after the try and catch block is executed or completed.

Syntax

```
Class finally dem
{
Static void pro x ( )
{
Try
{
System.out.println (“in proc x”);
Throw new runtimeException (“demo”);
}
Finally
{
System.out.println (“pro x finally”);
}
}
```

Event handling

Event

It shows the change in state of a source.

Event sources

It is an object generating an event.

Syntax

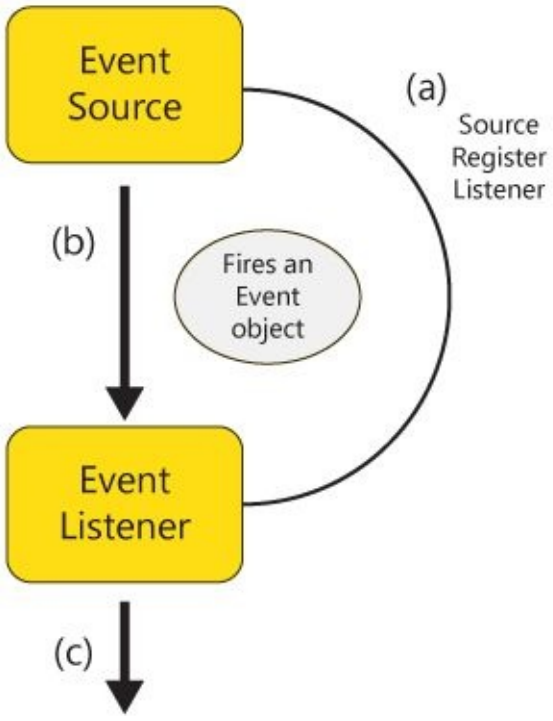
Public void add TypeListener (TypeListner l)

For example

- *Addkeylistner ( )* - to register a keyboard event.
- *Addmousemotionlistner ( )* - to register a mouse motion listener.

Action Event Class

This happens when button is pressed. Double clicking of the item or any item in the menu is selected.



Event occurrence in java

Key event class

This generated from the key input of the keyboard and there are three types of key events.

1. KEY_PRESSED

2. KEY_RELEASED
3. KEY_TYPED.

Mouse event class

There are eight types of mouse event classes.

1. MOUSE_DRAGGED
2. MOUSE_CLICKED
3. MOUSE_ENTERED
4. MOUSE_EXITED
5. MOUSE_MOVED
6. MOUSE_PRESSED
7. MOUSE_RELEASED
8. MOUSE_WHEELED

Action listener interface

It is used to define the actions these are all performed which are invoked at the call of an event. It is used to know the reaction of the method.

The Applet class

It contains several methods which are useful for java execution and detailed control. The Applet initialization and termination are as follow.

For starting

init( )- Used to initialize method it is the first method.

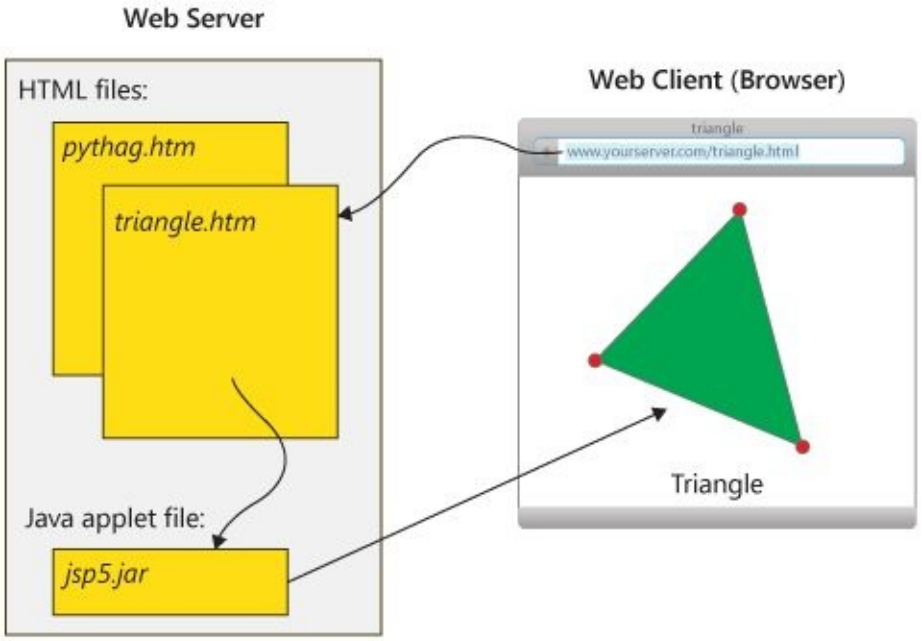
start( )- After initialization method this method comes and its function is to restart or display on screen.

paint( )- In this the output of the applet is redrawn each time it is called.

For stopping

stop( )- It is to leave the browser which is opened and contains applet.

destroy( )- It is for the applet once which has to removed from the memory.



Applet generation

HTML applet tag

<APPLET

[CODEBASE= codebaseurl]

CODE=applet file

[ALT=alternatetext]

[NAME=appletinstancename]

WIDTH=pixels HEIGHT=pixels

[ALIGN=alignment]

[VSPACE=pixels] [HSPACE=pixels]

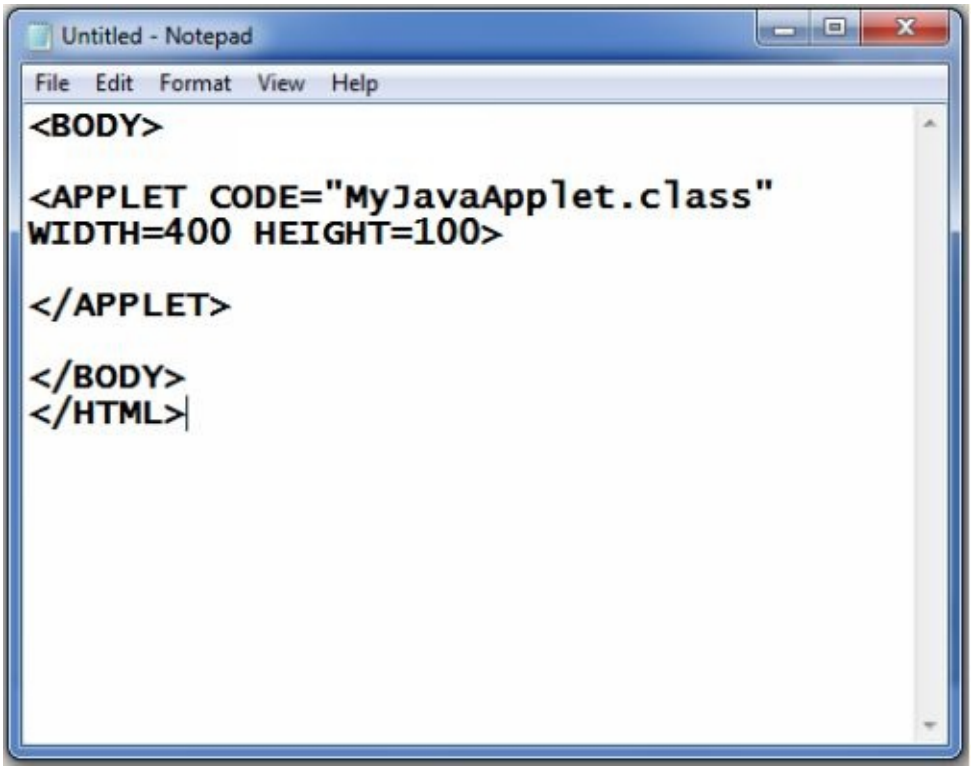
>

[PARANAME=attribute name VALUE= attributevalue>]

…

[HTML displayed in absence of java]

</APPLET>



HTML Applet tag

The Audioclip interface

play( )- To begin the audio.

stop( )- To stop the audio.

loop( )- To play the loop without break.

The Applet sub interface

It provides a link through which an Applet and the browser can communicate.

Knowledge flow: more eBooks and Apps

- *Get more Play Books*
- *Get more Kindle eBooks*
- *Get more apps from Google Play store*
- *Get more apps from Amazon app store*