
JAVA

JDK 1.02

WWW : World Wide Wep

www : Client Server

-- ----->

<-- ---

client : browser

Java
www가 가 (가)

Java : java application, java applet

editor source java compiler

java :

byte code

Java application : browser

Java applet : www browser

Java

, , , , , ,


```
void GoodVariableExample() {
    int    some_variable = 5;

    system.out.println( some_variable + 1 );
}
```

Overloaded Methods

An overloaded method : a method that you can call with different sets of parameter lists.

example 1

```
class ProgressBar {
    float percent_done;

    void SetProgress( float new_percent_done ) {
        percent_done = new_percent_done;
    }
    void SetProgress( int items_done, int items_total )    {
        percent_done = items_done / items_total;
    }
}
```

Member Variables

use classes to create container objects that do not contain any code.

example 1

```
class DateObject {
    int Day;
    int Month;
    int Year;
}
```

Final Variables as Constants

want to refer to a value by name, but don't want to change the value in your code.

example 1

```
class DateObject {
    int Day = 14;
    int Month = 4;
    int Year = 68;
    final int Max_Month = 12;    // can't be changed
}
```

Constructors

ensure that a newly created object has been properly set up.

function that gets called when an object is created


```

class SpaceAlien {
    boolean alive;
    static int alien_count = 0;

    void SpaceAlien() {
        alive = true;
        alien_count ++;
    }
    void Blast() {
        if ( alive ) {
            alive = false;
            alien_count --;
        }
    }
}

```

Static Methods

1. `class` : method, object
 가 .
 example 1

```

class MathStuff {
    static int SquareIt( int x ) {
        return x * x;
    }
    static int HalfIt( int x ) {
        return x / 2;
    }
}

class SomeClass {
    void SomeMethod () {
        int i = MathStuff.SquareIt( 4 );
        int j = MathStuff.HalfIt( I );
        System.out.println( j );
    }
}

```

2. static , 가
 example 1

```

class Users {
    static String user_list[] = new String[3];
    static {
        user_list[0] = "Ed";
        user_list[1] = "Joe";
        user_list[2] = "Scott";
        System.out.println( "Users loaded and initialized." );
    }
}

```

Creating Objects

"new" operator : object , parameter list parameter
 constructor

example 1

```

DateObject theDate;
theDate = new DateObject(); or the Date = new DateObject( 4, 14, 66 );

```

Object References

example 1

```

theDate.Year = 1995;

```

example 2

```

Lamp theLamp = new Lamp();
theLamp.SetState( true );

```

this

a reference to the current object

example 1

```

class LightBulb {
    Lamp myLamp;

    LightBulb( Lamp theLamp ) {
        myLamp = theLamp;
    }

    void PrintState () {
        if ( myLamp.power_on ) {
            System.out.println( "Lamp is on" );
        } else {
            System.out.println( "Lamp is off" );
        }
    }
}

```

```

class Lamp {
    boolean power_on;
    LightBulb myLightBulb;

    lamp () {
        myLightBulb = new LightBulb( this );
    }

    Lamp( boolean power_on ) {
        this.power_on = power_on;
        myLightBulb = new LightBulb( this );
    }

    void SetState( boolean on_state ) {
        power_on = on_state;
    }
}

```

null

the reference is invalid

example 1

```

if ( myLightBulb != null ) {
    System.out.println( "LightBulb exists" );
}

```

Inheritance

Subclassing

the process by which you derive one class from another

example 1

```

class Feline {
    boolean hungry = true;
    void speak () {
    }
    void call () {
        System.out.println( "Here kitty, kitty, kitty..." );
        if ( hungry ) speak();
    }
}

class HouseCat extends Feline {
    void speack() {
        System.out.println( "Meow!" );
    }
}

```

Final Classes

A class that can't be subclassed

example 1

```
final class HouseCat extends Feline {  
...  
}
```

Base Classes and Object References

example 1

```
Feline theCat;  
theCat = new HouseCat();  
theCat.speak();
```

Polymorphism

example 1

```
class CatWorld {  
    Feline cat_list[] = new Feline[3];  
  
    void set_up_cat_list() {  
        cat_list[0] = new HouseCat();  
        cat_list[1] = new Lion();  
        cat_list[2] = new HouseCat();  
    }  
  
    void speak_all() {  
        for ( int i = 0; i < cat_list.length; i++ ) {  
            Feline theCat = cat_list[i];  
            theCat.speak();  
        }  
    }  
}
```

```
*      object      "Object" class      subclass      "Object" class      method  
      . (      : clone(), toString() ... )
```

Casting between Class Types

example 1

```
HouseCat theCat;  
theCat = new HouseCat();
```

```
Feline theFeline;  
theFeline = theCat;           // object
```

```
HouseCat aCat;  
aCat = (HouseCat) theFeline;
```

```
*      class      class      cat      (      : HouseCat theCat = (HouseCat) new Feline() )  
      subclass      base class      가      ..
```

Overriding Methods
parent class method name, return type, parameter 가 method override
.

```
class Feline {
...
    void speak() {
        }
...
}

class HouseCat extends Feline {
    void speak() {
        System.out.println( "Meow!" );
    }
}
```

The Super Variable and Overridden Methods
"super" : parent class method
example 1

```
class HouseCat extends Feline {
    void speak() {
        System.out.println( "Meow!" );
    }
}

class MagicCat extends HouseCat {
    boolean people_present;

    void speak() {
        if ( people_present )
            super.speak();
        else
            System.out.println( "Hello" );
    }
}
```

example 2

```
class class_A {
    int value;

    class_A () {
        System.out.println( "class_A default constructor" );
    }
    class_A( int a ) {
```

```

        value = a;
    }
}

class class_B extends class_A {
    class_B( int a ) {
        super( a );
        System.out.println( "class_B constructed" );
    }
}

```

Final Methods

subclassed

example 1

```

final boolean GetLampState() {
    return power_on;
}

```

Abstract Methods and Classes

subclassed

abstract methods 가 class abstract class , "new"
 가 .
 class base class

example 1

```

abstract class Feline {
    boolean hungry = true;
    abstract void speak();
    void call() {
        if ( hungry ) speak();
    }
}

```

Encapsulation

object

program

- 1) private , method midpoint
- 2) private method midpoint ,
- 3) method SetStartPoint SetEndPoint

example 1

```

class Line {
    private int startpoint_x;
    private int startpoint_y;
}

```

```

private int endpoint_x;
private int endpoint_y;
private int midpoint_x;
private int midpoint_y;

void SetStartPoint( int x, int y ) {
    startpoint_x = x;
    startpoint_y = y;
    CalcMidPoint();
}
void SetEndPoint( int x, int y ) {
    endpoint_x = x;
    endpoint_y = y;
    CalcMidPoint();
}
private void CalcMidPoint() {
    midpoint_x = startpoint_x + (endpoint_x - startpoint_x) / 2;
    midpoint_y = startpoint_y + (endpoint_y - startpoint_y) / 2;
}
}

```

Access Modifier (visibility)

package 가

Public 가

source file public class가 , public class

JAVA applet Java.Applet subclass public class가

Private methods class

method subclass , override .
class .

Protected methods class
subclass .
class .

Interface

Interface and Multiple Inheritance

Interface : abstract, static and final

class superclass interface

```

class      interface      implement      ,      interface      method      가
.
example 1
interface Drawable () {           // method
    void drawMe( int x, int y );           //           abstract
}

class SpaceAlien implements Drawable {
    ...
    public void drawMe( int x, int y ) { // drawMe method가
        // Code for drawing the space alien
    }
    ...
}

class HouseCat extends Feline implements Drawble {
    ...
    public void drawMe( int x, int y ) { // drawMe method가
        // Code for drawing the house cat
    }
    ...
}

```

Using Interface as Class Type

example 1

```

HouseCat theCat = new HouseCat();
SpaceAlien theAlien = new SpaceAlien();

```

```

Drawable theDrawableObject;

```

```

theDrawableObject = theCat;
theDrawableObject.drawMe( 10, 10 );

```

```

theDrawableObject = theAlien;
theDrawableObject.drawMe( 10, 10 );

```

example 2

```

Drawable[] theDrawList = new Drawable[3];

```

```

theDrawList[0] = new HouseCat();
theDrawList[1] = new SpaceAlien();
theDrawList[2] = new HouseCat();

```

```

for ( int i = 0; i < theDrawList.length; i++ )
    theDrawList[i].drawMe( x, y );

```

Accessing Interface Variables

final and static

example 1

```
interface Drawable {
    int draw_area_top = 10;
    int draw_area_left = 10;
    int draw_area_bottom = 100;
    int draw_area_right = 200;

    void drawMe( int x, int y );
}
```

"Subclassing" Interfaces

example 1

```
interface Draw_Constants {
    int draw_area_top = 10;
    int draw_area_left = 10;
    int draw_area_bottom = 100;
    int draw_area_right = 200;
}

interface Drawable extends Draw_Constants {
    void drawMe( int x, int y );
}
```

Packages

colletctions of classes

Using Packages

1. use the fully qualified class name
java.util.Date theDate = new java.util.Date();
System.out.println(theDate);

2. `class`
import java.util.Date
Date theDate = new Date();
System.out.println(theDate);

* `java.lang` import `.`

Compiling with Packages

Package class CLASSPATH 가
: import java.lang.Math , lang directory Math.class .

Creating Package

MyPackage directory MyClass.java

```
package MyPackage;
```

```
public class MyClass {
    public void test() {
        System.out.println( "Test" );
    }
}
```

```
PackageTest.java
import MyPackage.MyClass;
```

```
class PackageTest {
    public void main ( String argv[] ) {
        MyClass theClass = new MyClass();
        theClass.test();
    }
}
```

file directory

classes

```
PackageTest.java
PackageTest.class
MyPackage
    MyClass.java
    MyClass.class
```

```
* java Dynamic binding MyClass.class PackageTest.class
, class loader .
```

Vector

resizable array class

example 1

```
Vector theVector = new Vector();
for ( int i = 0; i < 3; i++ ) {
    Integer newInteger = new Integer( i );
    theVector.addElement( newInteger );
}
```

example 2

```
Vector nameVector = new Vector();
String current_name;

current_name = "Ed";
nameVector.addElement( current_name );
```

```
* Vector object : insertElementAt( Object obj, int index );
```

```

* Vector element : removeElement( object obj );
* element : removeElementAt( int index );
* element : removeAllElement();
* Vector element : Object elementAt( int index );
* Vector element : void setElementAt( Object obj, int index );
* Vector : boolean contains( Object obj );
* Vector : int indexOf( Object elem ); // - 1
* : int indexOf( Object elem, int index );
* backward : int lastIndexOf( Object elem ); int lastIndexOf( Object elem, int
index );
* : Object firstElement(), lastElement()

```

Hashtables
object key

example 1

```

HashTable g = new HashTable();
String def = "A class derived from another class."; search;
g.put( "subclass", def );
Def = "A class from which ohter classes arre derived.";
g.put( "superclass", def );
search = (String) g.get( "subclass" );

* object : Object put( Object key, Object value );
* object : Object remove( Object key );
* Object get( Object key ); boolean containsKey( Object key );

```

Handling Exception

```

가 exception throw catch exception
handler jump
example 1
void BadMethod() {
    String myString = null;
    try {
        if ( myString.equals( "Test" ) )
            System.out.println( "String matched" );
    } catch ( NullPointerException e ) {
        System.out.println( "A null pointer exception ocured" );
    } catch ( Exception e ) {
        System.out.println( "An exception occurred" );
    }
    System.out.println( "Method done" );
}

```

Elements of Exception Handling

Throw

exception 가

throw reference-to-Throwable-subclass; : throw new ArithmeticException();

Try, Catch

try statement;

catch (Throwable-subclass e) statement;

finally

exception ,

try statement;

catch (Throwable-subclass e) statement;

finally statement;

Thread

multithreaded class 가

1. Create a class that extends the Thread class
2. Create a class that implements the Runnable interface

1. Creating Threads

```
class className extends Thread {  
    public void run() {  
        // Thread body of execution  
    }  
}
```

```
className a = new className();
```

```
a.start(); // run() ,
```

2. class className implements Runnable

```
public void run() {  
    ...  
}
```

```
className instanceName = new className();
```

```
new Thread(instanceName).start();
```

java.lang package

import 가

language classes method static member new

the Object class

class top - superclass

```
protected Object clone() : Object
```

```
Rectangle rect = new Rectangle( 10, 20 );
Rectangle rectcopy = rect.clone();
```

```
protected void copy( Object src ) : Object
Rectangle rect1 = new Rectangle( 10, 20 );
Rectangle rect2 = new Rectangle( 20, 20 );
rect1.copy( rect2 )           // rect1  20, 20
```

```
public final Class getClass()           : return the class definition object class
```

The Type Wrapper

```
class Object
public ClassType( type )
Integer I = new Integer( 1 );
Character ch = new Character( 'a' );
Float flt = new Float( 1.234 );
```

```
public type typeValue()
int x = i.intValue();
char y = ch.charValue();
float z = flt.floatValue();
```

```
public String toString()
System.out.println( i.toString() ); .....
```

```
public boolean equals( Object obj )
Integer j = new Integer( 2 );
boolean bool = j.equals( i );
```

The Boolean Type Wrapper

```
bool = Boolean.TRUE, Boolean.FALSE
```

The Character Type Wrapper

```
equal()                method  static
public static boolean isLowerCase( char ch );
public static boolean isUpperCase( char ch );
public static int digit( char ch, int radix ); // radix      (2,8,10,16).
           int x = Character.digit( 'a', 10 );           // x = 10
public static boolean isDigit( char ch );           // ch가 0~9
public static char forDigit( int digit, int radix );
public static char toLowerCase( char ch );
public static char toUpperCase( char ch );
```

Type Wrappers for Numbers

```
public int intValue();
public long longValue();
```

```
public float floatValue();
public double doubleValue();
```

```
Double d = new Double( 1.234 );
int x = d.intValue();
```

```
int I = Integer.valueOf( "1234" ).intValue();
```

The Float Type Wrapper

```
public final static float E // e = e.718281...
public final static float PI // pi = 2.141592...
```

```
float f = Float.E;
```

Not a Number

```
float f = 0, j = 1000;
j = j / f;
boolean bool = isInfinite( j ); // bool is true
```

The Integer Type Wrapper

```
public Integer( String s );
public static String toString( int I, int radix );
public static int parseInt( String s, int radix );
public static Integer valueOf( String s, int radix );
```

The String Classes

가 String classes

1. String : constant
2. StringBuffer :

Using String

```
String s = "Hello, World!";
String s = new String( char value[] );
public String( char value[], int offset, int count );
string :
public String( byte ascii[], int hibyte ); // UNICODE upper 8 bit
public String( byte ascii[], int hibyte, int offset, int count );
```

```
public int length();
public char charAt( int index ) : character return
public boolean equals( Object o ) : String objects
public int compareTo( String s ) : String 0 ...
public boolean regionMatches( int toffset, String other, int ooffset, int len )
: other string ooffset len region match
public boolean startsWith( String prefix ) : , prefix
public boolean endsWith( String suffix ) : string
```

```

public int indexOf( String str )      : str , - 1
public String substring( int beginIndex, int endIndex ) : beginIndex endIndex string
public String concat( String str );
public String toLowerCase()
public String toUpperCase()
public char toCharArray() : string new array of characters
public static String valueOf( type variable )

```

The StringBuffer Class

3 constructor

```

public StringBuffer() : empty StringBuffer
public StringBuffer( int length );
public StringBuffer( String str );

```

```

StringBuffer str = new StringBuffer();
System.out.println( "Enter Width" );
while ( ( ch = System.in.read() ) != '\n' ) {
    str.append( ch );
}

```

```

public int length()
public int capacity() : 가
public synchronized StringBuffer append( type variable )
public synchronized StringBuffer insert( int offset, type variable )
public synchronized StringBuffer insertChar( int offset, char ch )
public synchronized String toString()

```

The System Class

System.method

```

public static InputStream in
public static PrintStream out
public static PrintStream err
System.in.read(), System.out.print(String), System.out.println(String)

public static int currentTime() // timed event ,
public static int freeMemory()
public static int totalMemory()
public static void exit( int status )
public static InputStream exein( String command ) // program command
    FileInputStream .
public static OutputStream exein( String command)
public static void exec( String command )
public static String getenv( String var )
public static String getCWD() // Current Working Directory
public static String getOSName()

```

The IO and Utility Class Libraries

```
java.io class      class
InputStream stdin = System.in
OutputStream stdout = System.out
```

```
InputStream
public abstract int read()
public int read( byte b[] )
    class input {
        public static void main( String args[] ) {
            throws java.io.IOException {
                byte buf[] = new byte[50];
                System.in.read( buf );
                String str = new String( buf, 0 );
                System.out.println( str );
            }
        }
    }
public int read( byte b[], int off, int len )
```

```
PrintStream
public void print( type variableName )
public void println( type variableName )
```

```
The file class
public File( String path )
c:\user\text\input.txt
File dir = new File( "c:\user\text" );
File dir = new file( "text" );
public File( String path, String name )
File input = new File( "text", "input.txt" );
public File( File dir, String name )
File input = new File( dir, "input.txt" );
```

```
The RandomAccessFile Class
public RandomAccessFile( String name, String mode ) // mode "r", "rw"...
public RandomAccessFile( File file, String mode )
public int length()
public void seek( int pos )
public int getFilePointer()
public void close()

finalize {
    fileName.close()
}
```

```

public int read()
public int read( byte b[] )
public int read( byte b[], int off, int len )
public final String readLine()
public final String readUTF()
public final boolean readBoolean()
public final byte readByte()
public final short readShort(), ...

public void write()
public void write( byte b[] )
public void write( byte b[], int off, int len )
public final void writeLine( String line )
public final void writeUTF( String utf )
public final void writeBoolean( boolean b )
public final void writeByte( byte b )
public final void writeChar( char c )
public final void writeInt( int i )
public final void writeLong( long l )
public final void writeBytes( String s )           // ASCII code

```

The Utility Class Library

The Date Class

```

public Date()
public Date( int year, int month, int date )
public Date( int year, int month, int date, int hrs, int min )
Date    1~31           0
Date today = new Date()
System.out.println( today.toString() ); System.out.println( today )

```

The Stack Class

```

public Stack()
public Object push( Object item )
public Object pop()
public Object peek()
public boolean empty()
public int search( Object o )

```

Using the applet and awt Class Libraries

```

applet가 browser load , init() method가 .
public void init() {
    resize( 25, 25 );
}

```

Getting Attributes

```

<param> tag          load
public class ClickAnimation extends Applet implements Runnable {
    Thread animationThread = null;
    protected int cycles = 0;

    public void init() {
        resize( 25, 25 );
        String cycleTemp = getParameter( "CYCLES" );
        if ( cycleTemp != null )
            cycles = Integer.valueOf(cycleTemp).intValue();
        ...
        for ( int I = 0; I < 10; I++ )
            frame[I] = getImage( getDocumentBase(), "frame "+I+".gif" );
    }
}

applet          init() {          , start() {
public void start() {
    animationThread = new Thread( this );
    animationThread.start();
}

```

```

import java.awt.*;
import java.applet.*;

```

```

public class ClickAnimation extends Applet implements Runnable {
    Thread animationThread = null;
    protected int cycles = 0, cur = 0;
    protected Image frame[] = new Image[10];

    public void init() {
        resize( 30, 30 );
        String cycleTemp = getParameter( "CYCLES" );
        if ( cycleTemp != null )
            cycles = Integer.valueOf(cycleTemp).intValue();
        for ( int I = 0; I < 10; I++ )
            frame[I] = getImage( getDocumentBase(), "frame "+I+".gif" );
    }
    public void start() {
        animationThread = new Thread( this );
        animationThread.start();
    }
    public void run() {
        for ( int I = 0; I < 10; I++ ) {
            for ( cur = 1; cur < 10; cur++ ) {
                repaint();
            }
        }
    }
}

```

```

        for ( int j = 0; j< 50000; j++ ) ;
    }
    cur = 0;
    repaint();
}
}
public void paint( Graphics g ) {
    g.drawImage( frame[cur], 0, 0, this );
}
}

```

Handling Mouse Events

```

public boolean mouseDown( Event e, int x, int y )
{
    animationThread = new Thread( this );
    animationThread.start();
    return true;
}

```

Dealing with Keystrokes

```

public boolean keyDown( Event e, int c ) {
    System.out.println( "keyDown" + (char) c );
    return true;
}

```

Java for the internet

The java.net Classes

ContentHandler, InetAddress, ServeSocket, Socket, SocketImpl, URL, URLConnectin, URLStreamHandler

The InetAddress Class

```

public String getHostName() // host name
public String getAddress() // host numerical address
public static InetAddress getLocalHost() // application local host (InetAddress
object )
public static synchronized InetAddress getByName( String host) host InetAddress
public static synchronized InetAddress[] getAllByName( String host )

```

```
import java.net.*;
```

```

public class getAddress extends java.applet.Applet
{
    static InetAddress addr;
    static InetAddress addrList[] = null;
    static String ipAddress;
}

```

```
public FtpClient( String host, int port )
public FtpClient()

public void openServer( String host )           // FTP server      connection  open
public void openServer( String host, int port )

open      , login
public void login( String user, String password )

public void cd( String remoteDirectory )
public void binary()
public void ascii()

public TelnetInputStream list()           //      directory  input stream
public TelnetInputStream get( String filename )
public TelnetOutputStream put( String filename )
```