

Class structure

fields/attributes/- properties	object Status
Methods/Behavio- rs/Functions	object behavior
comments (3 types)	//single line /* multiple*/ /**java doc */
classes vs files	one file have one public class one file could have more classes

Main()

JVM hook point to process, program entry point

public static void main(array input
String[] args){}

public static void main(varargs list
String... args){}

args[0] is the first separated by
argument space

javac -cp ".,;:..." > MyClass.class
MyClass.java

java -cp "... " MyClass args[0]="San
"San Diego"

java -cp "... " MyClas- X MyClass
s.class

-cp ".,; c:\tmp\my*" sub directory not
included by *

variable arguments: array arguments with
variable size. int... num is an array with
variable size. Only the last parameter could
be varargs.

Creating Objects

Constructor initialize instance fields

name=Class public always
name

no return type not void either

new Constructor()

default constr- do nothing constructor
uctor

getter

setter

code block { }

**initializer
block** code block out of method
block

initialization field declaration line

order

static initializer block

instance initializer block

constructor

Packages and import

packages=organizer avoid naming
for easy to find/use conflict;control
types(class, inter- access to code(p-
face,enum); ackage private)

Syntax

import java.nio.*; X only subpackage
here

import java.nio.fil- X can not import
es.Paths.*; fields or methos

import java.nio.fil- import class
es.Paths;

import java.nio.files.*; import classes

import java.nio.files.; X

Packages and import (cont)

import java.lang.*; OK, redundant

Naming conflicts

import java.sql.*; Date is ambiguous
import java.util.*

Class import precedent *

import java.sql.Date; import collides
import java.util.Date;

use full qualified class name to avoid
naming conflicts

Create package first line in file

javac pkg/ClassA.java pkg/ClassB.java

java pkg.ClassA.class X usepkg.ClassA

Classpath

java -cp ".,; c:/java/lib/*; ..." pkg.ClassA

c:/java/lib/* jar or subdir by
import/pkg
statement

c:/java/lib all .class file in the
dir

-cp will ignore environment class_path
variable

import static

import a.b.X.*; direct access all
static member

if access static by ref, need to import the
class, instead of static import

java import only make compiling a little
longer, not make program larger.

JAR file

Java ole format zip file with optional
Archive .jar ext

Content class files+resource files

why jar?

C

By **Jianmin Feng** (taotao)
cheatography.com/taotao/

Not published yet.

Last updated 9th May, 2019.

Page 1 of 3.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

JAR file (cont)

Security _digital signing	compression
package versioning	portable
Create jar	jar cf jar-file input-file(s)/directories
	_export
classpath=c:\lib\Car.jar	
classpath=c:\lib*	all jars
classpath=c:\lib	all .class files

Primitive types

primitives	byte	literal	default
boolean	bit	true	false
byte	1	123	0
short	2	123	0
int	4	123	0
long	8	123L	0
float	4	123.5f	0.0f
double	8	123.5	0.0
char	2	'a'	"

Assignment of literal

Long max=31- 23456789	X	Long max=31- 23456789L	OK
byte b =123;	OK	byte b=256; x	
float f=123.5;	X	float f=123.5f;	OK

default 123 is int, 123.5 is double
 default value is for instance variable, local variable has no default, need to be initialized explicitly before use it
 char is treated as integer number in java, could use ++ -- and conversion between int, byte, short.

Number format conversion

binary	ob11	1,2,4,8	3
octal	o17	1,8,64	15
decimal	56	1,10,100	56
hexadecimal	ox1f	1,16,256	31

System.out.println(ox1f) will display in decimal 31.

_ used for all number formats

_123 is a valid identifier

underscore for easy read, _ could not anywhere but 4 locations: beginning, end, before or after decimal point.

123 _45_

Reference types

Reference type reference an object, hold the memory location of the object it reference to.

A reference also called a pointer

The default value for a instance reference type is null

local reference variable must be initialized to a null or object

Reference vs primitives

	References	Primitives
data	obj memory addr	value
default value	null	0, false, 0.0f, 0.0, NUL

java strong typed, can not assign 1 to boolean;

null is an object, can print it ("null"), cannot call method from it - NullPointerException.

Object vs Primitive

primitive has only one piece of state information, absolutely no behavior

Object bundled with one or multiple states and/or behavior

Object on heap, ref by ref type

Stack vs Heap

Stack	Heap
small	large
Ref variables	objects
live-span within scope	live or ready for gc
primitive value/member address	object state

Declare and initializing variables

variable name	Starts lower letters \$ _
	my contains numbers
package, keyword	lower letters
Class	Initcap
Constant	all caps
int a, string s;	X

Default initialization of variables

local variable	must be explicitly initialized
instance/class variables	false, 0, 0.0, "\u0000"(-NUL), null
boolean	false
byte, short, int, long	0
float double	0.0
char	"\u0000"(NUL)

Default initialization of variables (cont)

reference type null

Variable scope

block multiple levels
scope

methods life time => from declaration
scope to end of block

instance life time => from declaration to
scope gc of obj

class static lifetime=> from declaration to
scope end of program

a variable declared in switch,if,for loop can
not used outside of them

Order of element in a class

package declaration

import statement

class declaration

field declaration

methods declaration

Garbage collection

automatically remove the referenced
handled obj from heap

could manual call just a hit, may not get
System.gc() -->- run, but if run, never
finalize() run twice

Obj available for set to null; change ref;
gc: unreferenced all ref out of scope

Object References

on heap on stack

diff in size same size (mem addr)

accessed by ref by variable name

Memory Leak

lazy code:leave create variable never
unused obj used
hanging

Benefit of Java

Object oriented

encaps- getter, setter + private
ulation

platform independent

robust auto mem mgn, gc

simple no pointer, no operator
overloading

secure run within JVM (sandbox)