Cheatography

Java Building Blocks (OCA) Cheat Sheet

by Jianmin Feng (taotao) via cheatography.com/79308/cs/19322/

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 prcess, program entry point

public static void main(String[] args){}	array input
public static void main(String... args){}	varargs list
args[0] is the first argument	separated by space
javac -cp ",;..;.." MyClass.java	> MyClass.class
java -cp "..." MyClass "San Diego"	args[0]="San
java -cp "..." MyClas-s.class	X MyClass
-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 name	public always
no return type	not void either
new Contructor()	
default constr-uctor	do nothing constructor
getter	
setter	
code block	{ }
initializer block	code block out of method block
initialization order	field declaration line
	static initializer block
	instance initializer block
	constructor

Packages and import

packages=organizer for easy to find/use types(class, interf-ace,enum);	avoid naming conflict;control access to code(p-ackage private)
Syntax	
import java.nio.*;	X only subpackage here
import java.nio.fil-es.Paths.*;	X can not import fields or methos
import java.nio.fil-es.Paths;	import class
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.*; import java.util.*	Date is ambiguous
Class import precedent *	
import java.sql.Date; import java.util.Date;	import collides
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 Archive	ole format zip file with optional .jar ext
Content	class files+resource files
why jar?	

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

Not published yet.
Last updated 9th May, 2019.

Cheatography

Java Building Blocks (OCA) Cheat Sheet

by Jianmin Feng (taotao) via cheatography.com/79308/cs/19322/

JAR file (cont)

Security _digital signning	compression
package versioning	portable
Create jar	jar cf jar-file imput-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 - NullPointException.

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 duble	0.0
char	'\u0000'(NUL)

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

Not published yet.
Last updated 9th May, 2019.

Default initialization of variables (cont)

reference type	null

Variable scope

block scope	multiple levels
methods scope	life time => from declaration to end of block
instance scope	life time =>from declaration to gc of obj
class static scope	lifetime=> from declaration to end of program

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

Order of element in a class

1. package declaration
2. import statement
3. class declaration
4. field declaration
5. methods declaration

Garbage collection

automatically handled	remove the referenced obj from heap
could manual call System.gc() -->- finalize()	just a hit, may not get run, but if run, never run twice
Obj available for gc: unreferenced	set to null; change ref; 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 unused obj hanging	create variable never used

Benefit of Java

Object oriented	
encaps-ulation	getter, setter + private
platform independent	
robust	auto mem mgn, gc
simple	no pointer, no operator overloading
secure	run within JVM (sandbox)

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

Not published yet.
Last updated 9th May, 2019.