Cheatography

OOv1 Cheat Sheet

by Philip Schmid (Higarigh) via cheatography.com/20304/cs/3111/

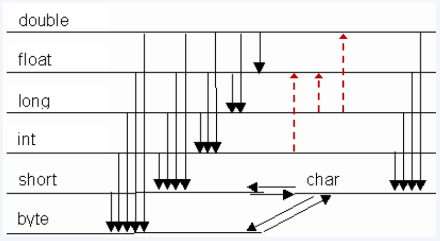
Java basics

`x = a ? b : c`	a true? x = b, a false? x = c
`0.1 + 0.1 == 0.3`	False, workaround: `Math.abs(0.1 + 0.1) < 0.2e6`
`a && b`	Only check b if a is true
`a || b`	Only check b if a is false
`0b11001`	binary, leading "0b"
`0x1e`	hexadecimal, leading "0x"
`010 != 10`	Leading 0 means octal
`'A' == 'a'`	False (case sensitive)
`'A' < 'B'`	True

Java primitive data types

boolean	true, false
char	16 bit, UTF-16
byte	8 bit, -128...127
short	16 bit, -32.768 ... 32.767
int	32 bit, -2^{31} to $+2^{31}-1$
long	64 bit, -2^{63} to $+2^{63}-1$, `long x = 100l;`
float	32 bit, `float x = 100f;`
double	64 bit, `double x = 100d;`

Java type casting



red arrows =implizit (probably information loss due inaccurate dataformat)
black arrows = explizit cast (heavy information loss possible --> developer)

Java interfaces

Methods in intefaces are implicitly public and abstract and can't be private.

Only constant variables are allowed: `public static final int HIGHWA Y_M IN_ SPEED = 60;` "public static final" optional.

Same named methods in basic interface and super interface must have the same return type.

Same named variables in basic interface and super interface can have different return types.

Default methods: Basic interface doesn't have to implement default methods. If one method is not overriden, it just takes the default method.

Java reference types

```
int[] x = new int[10];
```
normal Array

```
int[][] m = new int[2][3];
```
2 dimensional array

```
Arrays.eq uals(a, b)
```
Compare array content

```
Arrays.de epE qua ls(a, b)
```
Compare x dimensional array

```
public enum Weekday{ MONDAY, ..., SUNDAY }
```
Enum

```
String a = " Pro g1";
```
new reference to String Object "Pro g1" already exists, otherwise create it)

```
String c = new String ("Pr og1 ");
```
new String-Object

```
a.equa ls(b)
```
Compare Strings

Java reference types (cont)

```
String lde rbu ilder = new Str
de r.t oSt ring();
```

Java equals() example

```
@Override
public boolean equals (Object
obj) {
    if (obj == null) {
        return false;
    } else if (getCl ass() !=
obj.ge tCl ass()) {
        return false;
    } else if (!supe r.e qua ls( -
obj)) {
        return false;
    } else {
        {return z})
        Student other = (Stude -
nt)obj; return regNumber ==
other.r eg Number;
    }
```

If equals() is changed, hashCode() has to be changed (x.equals(y) -> x.hashCode() == y.hashCode()

Keywords

public	can be seen by all that imports this package
protected	can be seen by all classes in this package and all subclasses of this
package	can be seen by all classes in this package
private	can be seen only by this class
static	only once for all instances of this class

By **Philip Schmid** (Higarigh)
cheatography.com/higarigh/

Published 25th January, 2015.
Last updated 12th May, 2016.

Cheatography

OOv1 Cheat Sheet

by Philip Schmid (Higarigh) via cheatography.com/20304/cs/3111/

Keywords (cont)

final	can only be defined once and not changed later. Class: no subclasses, Method: no overriding
static final	Means this is a constant

Javadoc

Start with /**	end with */	each line *
@author name	author	class / interface
@version number	version	class / interface
@param name description	parameter	method
@return description	returnvalue	method
@throws/@exception type description	potential exception	method
@deprecated description	deprecated (outdated)	method

Java hashcode() example

```
public int hashCode() {
        return firstN ame.ha - shC ode() + 31 * surNam e.h - ash Code();
}
```

Java compareTo example

```
class Person implements Comparable<Person> {
  private String firstName, lastName;
  // Constr uctor…
  @Override
  public int compar eTo (Person other) {
        int c = compar eSt rin - gs( las tName, other.l as - tName);
>     if (c != 0) { return c; }
    else { return compareStrings(firstName, other.firstName); }
 }
private int compareStrings(String a, String b) {
  if (a == null) { return b == null ? 0 : 1; }
  else { return a.compareTo(b); }
 }
}
```

Java collections

```
ArrayL ist <Ob jec t> a1 = new ArrayL ist <>();
Linked Lis t<O bje ct> l1 = new Linked Lis - t<>();
Set<St rin g> s1= new TreeSe t<>(); or Set<St rin g> s2= new HashSe t<>();
```

Java collections (cont)

```
Map<In teger, Object> m1= new Hash ); or Map<In teger, Object> m2= new p<>();
Iterat or< Str ing> it= m1.ite rat
```

Java inheritance

```
Vehicle v1 = new Car();
```

get(int), set(int, " OO"), Object o = new Vehicle(); Vehicle), size() ;

add("IC Th"), add(int, "

add("Te st"), remove ("Te always efficient. HashSet = unsort

```
Vehicle v = new Vehicle; Car c = (
if(v instanceof Car) { Car c = (Ca
super.v ar iable
((Supe rSu per Cla ss) thi s).v ar
```

Dynamic dispatch: Methods: from dynamic typ and variables from static type.

OOv1 Cheat Sheet

by Philip Schmid (Higarigh) via cheatography.com/20304/cs/3111/

Java Lambdas / Stream API

```
Collec tio ns.s or t(p eople, (p1, p2) -> p1.getAge().compareTo(p2.getAge()));

Collec tio ns.s or t(p eople, (p1, p2) -> p1.getName().compareTo(p2.getName()));

people.st rea m().fi lter(p -> p.getAge()>18).map(p -> p.getName()).sorted() .forEa ch( Sys tem.ou t:: pri ntln);

people.st ream() .filter(p -> p.getL ast Name().sta rtsWi th("A")) .forEa ch( Sys tem.ou t:: pri ntln);

Random random= newRan dom (4711); Stream<Int eger> ran dom Int = random.ints(10, 0, 100).boxed();

List<P ers on> list= people Str eam.co llect( Col lec tors.t oLi st());
```

Java Lambdas / Stream API (cont)

```
Per son[] arr = people.toArray(new Person[people.length]);
```

Possible Stream API operations: filter (Predicate), map(Function), mapToInt(ToIntFunction), mapToDouble(ToDoubleFunction), flatMap(Function), distinct(), sorted(), sorted(Comparator), peek(Consumer), limit(long n), skip(long n), count(), min(), max(), average(), sum()

Comparator & Methodreference

Inferface used to compare 2 Objects(before you used lamdas).

Contains the method `public int compare(T o1, T o2)` which you need to override. Returns positiv number if o1 is bigger then o2 and negative if oppisite 0 means that they are equal

Instead of a comparator use methodreference eg `Person Comparator<Person> comparator = Person::compareByName;`

Nested class

Use this if a class is only used in another class

No seperate classfile

The inner class can use all members of the outer class (this include private members)

Instantiation from outside eg `Polygon.Point myPolygon = myPolygon.new Point();`

Can be declared in a method -> All variables from outside are getting final eg `int getMaxSpeed() {return 300; } }; return newSuperClass();`

Java own exception class

```
public class MyExceptionExtendsException {
  private static final long serialVersionUID = 1L;
  public MyException( String msg) {
    super( msg);
  }
}
```

Java package import conflict order

1. own class (inc. nested class)
2. single type imports -> `import p2.A;`
3. type in own package -> `package p1; class X`
4. Import on demand -> `import p2.*;`

Java regex

`1*`	0 to *
`1+`	1 to *
`1{2,5}`	min 2 max 5 (11..111
`1{2,}`	min 2 max * (11, 111,
`1{3}`	exactly 111
`-?1`	-1 or 1, "-" is optional
`Mo|Di`	Or
`[a-z]`	any letter from a to z
`[a-zA-Z]`	any letter from a to z
`\s`	whitespace
`.`	anything except newli
`[^abc]`	anything except a, b c
`$`	end of string
`\d`	any digit
`\D`	not digit
`(?< Gro up1 >REGEX)`	name capture group - `r.g rou p("G rou`

Example: Check daytime: `([0-1] ?[0 -9] |2[ 0-3 ]): [0- 5][0-9]`

OOv1 Cheat Sheet

by Philip Schmid (Higarigh) via cheatography.com/20304/cs/3111/

Java regex code example

```
String input =
scanner.nextLine();
Pattern pattern = Patter n.c -
omp ile ("([ 0-2 ]?[ 0-9 ]): -
([0 -5] [0- 9]) ");
Matcher matcher = patter n.m -
atc her (in put);
if (match er.m at ches()) {
    String hoursPart = matche -
r.g rou p(1);
    String minute sPart =
matche r.g rou p(2);
    Sys tem.ou t.p rin tln (..);
}
```

Java JUnit

Java JUnit examples

```
@Test
public voi dte stP rim e_2() {
    ass ert Tru e("2 isprim e",
utils.i sP rim e(2));
}
```

Java generics

Example: `class Node<T extends Number & Serial iza ble >{ … } Node<I nte ger> undefi nedNode = new Node<I nte ger>( );` can add different Interfaces with & to ensure other functionality like serializable

Wildcard type: `Node<?> undefi ned Node;`) and write (`.setVa lue(X)`) is allowed

static variables with generics NOT allowed eg static T maxSpeed;

Generic Method: `public <T> T majority(T x, T y, T z) { if(x.e qua ls(y)) { r } return null; }` Call: `Double d = test.< Dou ble >ma jor ity (1.0, 3.141,` types of argument)

Rawtype: like you would insert Object -> you need to down cast the elements. e.g: `Node n; //`

Serializable

Is a marker interface (is empty, just says that this class supports it)

Use it to say the developer that he can serialize objects of this class, which means he can write then in a bitecode and export them. Always serialize all the objects contained in the mainobject

Serializable (cont)

Use serialVersionUID to identify your class (nc

Example: `Output Stream fos= new File e rson); }`

Example: `InputS tream fis= new FileI r ea dOb ject(); … }`

Java clone() method

```
public Department clone() throws
Exception {
    Dep artment d = new Depart -
ment();
    d.name = name;
    d.people = people;
    d.s ubD epa rtments = new
ArrayL ist <>();
    for (Depar tment subD :
subDep art ments) {
        d.s ubD epa rtm ent s.a -
dd( sub D.c lon e());
    }
    return d;
}
```

`assert Equ als (ex pected, actual)`	actual «equals» expected
`assert Sam e(e xpe cte d,a ctual)`	actual== expected (only reference compar-ation)
`assert Not Sam e(e xpe cted, actual)`	expected != actual (only reference compar-ation)
`assert Tru e(c ond ition)`	condition
`assert Fal se( con dition)`	!condition
`assert Nul l(v alue)`	value== null
`assert Not Nul l(v alue)`	value!= null
`fail()`	everytime false
`@Test( tim eout= 5000)`	set test timeout
`@Test( exp ected= Illega lAr gum ent Exc ept io n.class)`	expect exception, if exception is thrown, test passes
`@Before public void setUp() { … }`	run this before each test
`@After public void tearDown() { … }`	run this after each test