Java + OOP concept Cheat Sheet

by Kunanon S (son9912) via cheatography.com/43384/cs/12920/

Hello World!

Start your Java day with Hello World program

```
public class HelloWorld {
    public static void main(S tring[] args) {
        // Prints " Hello, World" to the terminal window.
        System.ou t.p rin tln ("Hello, World");
    }
}
```

When you want to run the program, choose this class as main class.

Run your code

Compile from single class up HelloWorld class

```
>_ javac HelloW orl d.java
>_ java HelloWorld
```

Compile from multiple classes and choose main class

```
>_ javac *.java
>_ java HelloWorld // HelloWorld is your preferred main cl
ass
```

Variables

Type	Default Value	Memory Allocation
byte	0	8 bits
short	0	16 bits
int	0	32 bits
long	0L	64 bits
float	0.0F	32 bits (decimal)
double	0.00D	64 bits (decimal)
boolean	False	varies on impliment
String	NULL	depends on character count
char	\u0000	16 bits (unicode)

Operators

Operand	What they do
=	Assign value
==	Check value/address similarity
>	More than
>=	More than or equals
>>>	Move bit to the right by
++	Increment by 1

inverse of these operands still working the same.

For example : != is not equal

Defining variable

Defining new variable attributes

```
int x = 12;
int x; // will be defined as 0
```

Define by creating new instances

```
String x = new String;
```

Type Casting (decreasing bit use)

Expanding data types will not require type casting, Narrowing does.

```
double x = 10; // Expanding data types
int y = (int) 10.222222; // Narrowing data types
```

Conditions

If statement

```
if (state ment) {}
```

If - else statement

```
if (state ment) {} else{}
```

Switch

```
switch (num) {
   case 1: doSome thi ng();
        break;
    def ault: doThis();
        break;
}
```

Loop

```
for (int i: someArray) {}
while (somet hing) {}
do {somet hing} while (true)
```

Prime number function

```
if (n < 2) { return false; }
for (int i=2; i <= n/i; i++)
   if (n%i == 0) return false;
return true;
```

returns a boolean

String Pool - Optimizations

String pool is created to make the same value string use the same address. By doing that, it will save memory and time for compiler to do stuff

Basic testing

```
String s1 = " Hello World";
String s2 = " Hello World;
```

Check it using "=="

```
System.ou t.p rin tln(s1 == s2
);
>_ True
```

"==" will check its address

Allocate a new address using `new`

```
String s1 = " Hello World";
String s2 = new String;
s2 = " Hello World";
System.ou t.p rin tln(s1 == s2);
>_ False
```

Allocate new address by changing its value

```
String s1 = " Hello World";
String s2 = " Hello World";
s2 = " Hello Thaila nd";
System.ou t.p rin tln(s1 == s2);
>_ False
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Naming Grammars

Naming should be regulated for easier recogition from others

Use Upper Camel Case for **classes**: `Veloc i tyR esp ons eWriter`

Use Lower Case for **packages**: `com.co mp a ny.p ro ject.ui`

Use Lower Camel Case for **variables**: `stud en tName`

Use Upper Case for **constants**: `MAX_PA RA M ETE R_COUNT = 100`

Use Camel Case for **enum class** names

Use Upper Case for **enum values**

Don't use '_' anywhere except constants and enum values (which are constants).

Receiving user input

There is normally 2 ways to receive user keyboard input

1. java.util.Scanner

```
Scanner x = new Scanne r(S yst em.in);
String inputS tring = x.next(); // for String type input
int inputI nteger = x.next Int(); // for Integer type inpu
t
```

2. String[] args *from public static void main()*

NOTE: args is already in a array. It can receives unlimited amount of arguments.

```
String inputS tring = args[0]; // for String type input
Int inputS tring = (int) args[0]; // for Integer type inpu
t
```

To use Scanner, importing Scanner library is required : `import java.O bje ct.S - canner`

All types of input can be received. (not just String or int)

Access Modifier

	PRIVATE	DEFAULT	PROTECTED	PUBLIC
Same class	Yes	Yes	Yes	Yes
Same package Subclass	No	Yes	Yes	Yes
Same package Non-subclass	No	Yes	Yes	Yes
Different package Subclass	No	No	Yes	Yes
Different package Non-subclass	No	No	No	Yes

- Java uses `<de fau lt>` modifier when not assigning any.

- `public` modifier allows same class access

- *Works in inherited class* means itself and the classes that inherit from it.

Attribute modifier

Attribute Type	Access Grants
Private	Allows only in class where variable belongs
Public	Allows any class to have this attribute
Static	Attribute that dependent on class (not object)
Final	Defined once. Does not allow any change/inheritance

Methods

Methods are fucking easy, dud.

`<mo d> <re tur n> mthdName (<a rgs >) { }`

Example:

```
public double getAge () {
    return someDo uble;
}
```

Constructor

Constructors allow you to create an object template. It consists of **complete procedures**.

Create a blank constructor to allow its extension classes to inherit this *super* constructor.

`<mo dif ier> Person () {}`

Constructor (cont)

But will be created automatically by not writing any constructor

Create an argument-defined constructor

```
<mo dif ier> Person (String nam e) {
    thi s.name = name;
}
```

Abstract Class

Abstract is a type of class but it **can consist of**

Create new abstract

`<ac ces s_m odi fie r> abstract d () {}`

Interface

Interface is different from constructor. It **consists of incomplete assignments**

Interface allows you to *make sure* that any inherited class can do the following methods. (It's like a contract to agree that this thing mus be able to do this shit.) The method is then completed in the class that implements it.

Creating a new interface

```
interface Bicycle {
   void speedUp (int increm ent);
}
----
class fuckBike implements Bicycle
   ...
   void speedUp (int increment) {
        speed += increment;
   }
   ...
}
```



Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Java + OOP concept Cheat Sheet

by Kunanon S (son9912) via cheatography.com/43384/cs/12920/

Encapsulation

Encapsulation allows individual methods to have different access modifier.
Creating *setters* and *getters* is one way to use encapsulation.
For example

```
private void setTim e(int hour, int minuite, int second)
{
this.hour = hour;
this.m inuite = minuite;
this.s econd = second;
}
```

Inheritance

Inheritance helps class to import the superclass' method.

Importing superclass

```
class HelloWorld extends Object {}
```

Normally, the class that does not inherit any class will inherit *Object* class.*

Class can **only inherit 1 class/abstract**

Importing Interface

```
class HelloWorld inherits Interf ace Anin
g {}
```

Class can **inherit unlimited amount of interface**

Overload

We use overload when you want different input to work differently, but remains the same name.

Example of Overload

```
public printe r(S tring x){}
public printe r(S tring x, String y){}
```

If the input is 2 string, it will go to the second method instead of first one.

But you cannot overload by using the same input type sequence.

For example

```
public printe r(S tring x){}
public printe r(S tring x, String y){} // confli
ct
public printe r(S tring y, String x){} // confli
ct
```

Overload (cont)

Java will access only the first one to be run, because it cannot determine the value.

Override

When you have inherit some of the class from parents, but you want to do something different. In override feature, all the subclass/-class object will use the newer method.

To make sure JDK knows what you are doing, type `@Override` in front of the public name. If the override is unsuccessful, JDK will returns error.

Example of overriden helloWorld() method :

Class Student

```
public void helloW orld(){
System.ou t.p rin tln ("He llo ");
}
```

Class GradStudent extends Student

```
@Override
public void helloW orld(){
System.ou t.p rin tln ("Hello Worl
d");
}
```

Rules of Overridden methods

1. Access modifier priority can only be narrower or same as superclass
2. There is the same name method in superclass / libraries

java.io.PrintStream

Print with new line

```
System.ou t.p rin tln ("Hello W
orld");
```

Print

```
System.ou t.p rin t("Hello Worl
d"); // confli
```

java.util.Scanner

Create a Scanner object

```
Scanner sc = new Scanne r(S yst
.in);
```

Accept input

```
double d = sc.nex tDo uble()
```

java.lang.Math

Methods	Usa
`Math.m ax( <va lue 1>, <va lu e 2>)`	Ret max valu
`Math.m in( <va lue 1>, <va lu e 2>)`	Ret min
`Math.a bs( <va lue >)`	Ret uns
`Math.p ow( <nu mbe r>, <ex po n ent>`	Ret a num
`Math.s qrt (<v alu e>)`	Ret root

java.lang.String

Find the length -> int

```
msg.le ngth()
```

To lower/uppercase -> String

```
msg.to Low erC ase()
msg.to Upp erC ase()
```

Replace a string -> String

```
msg.re pla ceA ll( String a, Str
```

Split string between delimeter -> array

```
msg.sp lit (String delimeter)
```

Start/end with -> boolean

```
msg.st art sWi th( String pre)
msg.en dsW ith (String post)
```

String format -> String

```
String.fo rma t(S tring format,
 args)
```



By **Kunanon S** (son9912)
cheatography.com/son9912/

Published 25th September, 2017.
Last updated 29th September, 2017.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

java.lang.String

Methods	Description
charAt(int index)	Returns the char value at the specified index
compareTo-(String otherS-tring)	Compare 2 strings lexicographically
concat(String str)	Concatenate specified string
endsWith(String suffix)	Test if the string ends with specified suffix
equals(String andObject)	Test if strings values are the same
toCharArray()	Convert string to character array
toLowerCase()	Convert string to lowercase
toUpperCase()	Convert string to uppercase
toString()	Convert things to string
valueOf(<value>)	Return the represent-ation of argument
length()	Return length of the string
replaceAll(String a, String b)	Replace string a to string b
split(String delimeter)	Split string between delimeter
startsWith(String prefix)	Test if string starts with specified prefix
format(String format, Object arg)	Format strings to the format given

There is many more in Java documents : https://docs.oracle.com/javase/9/docs/api/-java/lang/String.html

java.util.Collection (CollectionAPI)

Provides ways to keep variables and access it faster

Ways to keep data

1. Set - Care about duplicity, not queue (eg. HashSet)
2. List - Care about queue, not duplicity (eg. LinkedList)
3. Map - Care about both queue and key duplicity (eg.HashMap)

Methods that will be included

```
boolean add(Object element);
boolean remove (Object element);
int size();
boolean isEmpty();
boolean contai ns( Object element );
Iterator Iterat or();
```

HashList - CollectionAPI

Method	Usability
void add (int index, Object element)	Add value to list
Object remove(int index)	Remove item #index from list
Object get(int index)	Retrieve item #index from list
void set(int index, Object element)	Set data to correspond #index
int indexOf(Object element)	Find the #index from element
ListIterator listIterator()	

It also includes all CollectionAPI methods

Create new HashList by using

```
List x = new HashLi st();
```

HashMap - CollectionAPI

Method	Usability

Collections

Create List of 1, 2, 3 on-the-fly

```
Arrays.as List(1, 2, 3)
```

Convert primitive array to Stream

```
Arrays.st rea m(p rim iti veA - rray)
```

Convert ArrayList to Stream

```
arrayL ist.st ream()
```

LinkedList - CollectionAPI

Create empty LinkedList of Integer

```
LinkedList myList = new Linked I er >t()
```

Create LinkedList with values in it

```
new Linked Lis t<> (Ar ray s.a s 3)))
```

Add an object to LinkedList

```
myList.ad d(50)
```



By **Kunanon S** (son9912)
cheatography.com/son9912/

Published 25th September, 2017.
Last updated 29th September, 2017.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com