Cheatography

Java17 Cheat Sheet

by lucamazzza via cheatography.com/194481/cs/40573/

Primitive Data types

Name	Size	Values
`byte`	1 B	-128 → 127
`short`	2 B	-32768 → 32767
`int`	4 B	~-2•10⁹ → ~2•10⁹
`long`	8 B	~-9•10¹⁵ → ~9•10¹⁵
`float`	4 B	6-7 decimal digits
`double`	8 B	15 decimal digits
`boolean`	1 b	`true` or `false`
`char`	2 B	Characters or ASCII

Comparison Operators

Equal	`x == y`	Not equal	`x != y`
Less	`x < y`	Less or equal	`x <= y`
Greater	`x > y`	Greater or equal	`x >= y`

String Methods

Usage	Description
`s.leng th();`	Length of a string
`s.char At(x);`	Extract the n^{th} character
`s.toUp per Case();`	Returns a copy of a string in ALL CAPS
`s.toLo wer Case();`	Returns a copy of a string in lowecase
`s.inde xOf(c);`	Returns the first occurring index of *c*
`s.repl ace (old, new );`	Search and Replace
`s.split(RegEx);`	Splits string into tokens
`s.equa ls(s2);`	Compares strings (true if equal)
`s.comp are To(s2);`	Returns "0" if equal, "+" if s > s2, "-" if s < s2

IF Statement

```
// conditional handler
if (condition) {
// pose a condition ...
} else if (other condition) {
// if first condition not met pose another one ...
} else {
// if second condition not met finally ...
}
```

FOR EACH Statement

```
// self-iterating loop for iterable objects
for (var: collection) {
  statements
}
```

FOR Statement

```
// structured iterator for non-iterable objects
for (int i = 0; i <  max; i++) {}
```

Continue

```
for (int i = 0; i <  max; i++) {
   // jump to the next iteration
    con tinue;
}
```

Break

```
while (true) {
   // exit early a loop
    break;
}
```

TRY-CATCH Statement

```
// exception handler
try {
// try to do something
} catch (exception e) {
// when you fail do...
} finally {
// when you are done do...
}
```

Enumerals

```
// "class" containing a group of constants
enum Name {
    NAME1,
    NAME2,
    NAME3
}
// constant are accessible per dot notation
// they are typed as the declared enum
Name var = Name.N AME1;
```

Object Example

```
public class Object {
    private <ty pe> attr1;
    public Object (<t ype> attr1) {
        thi s.attr1 = attr1;
   }
    public void method() {
       // ...
   }
}
```

Interfaces

```
// An interface manages accessibility of object
// it is an abstract object
public interface Intrf {
    String const = " CIA O";
    public void method1();
}
public class Obj implements Intrf {
    public void Obj() { }
    @ov erride
    public void method1() {}
}
```

Parent Class

```
// parent class
public class Parent implements Intrf {}
// child class, inherits Parents charac ter istics
public class Child extends Parent {}
```

ArrayList Methods

Usage	Description
`list.add(item);`	Add *item* to the list
`list.get(n);`	Return the n^{th} item
`list.s ize();`	Return number of item
`list.r emove(n);`	Remove the n^{th} item
`list.set(n, value);`	Put at the n^{th} position value

Non-Primitive Data Types

Name	Declaration
String	`String name = " Luc a";`
Array	`type[] = {x, y};`
Class	`access class Obj { ... }`
Interface	`public interface ObjInt erface { ... }`

Java17 Cheat Sheet

by lucamazzza via cheatography.com/194481/cs/40573/

Boolean Operators

NOT	`! x`
AND	`x && y`
OR	`x || y`
XOR	`x ^ y`

Arithmetic Operators

Addition	`x + y`
Subtraction	`x - y`
Multiplication	`x * y`
Division	`x / y`
Increment	`x++  ++x`
Decrement	`x--  --x`
Modulus	`x % y`

WHILE Statement

```
// pre-conditioned iterator ...
while (condition) {}
```

Scanner

```
import java.utils.Scanner;
public class Obj{
    public static void main (String[] args) {
            Scanner input = new Scanne r(S yst -
em.in);
            String str = input.n ex tLi ne();
           int i = input.n ex tInt();
            inp ut.c lo se();
   }
}
```

DO-WHILE Statement

```
// post-conditional iterator
do { } while (condition);
```

SWITCH Statement

```
// case-oriented condition handler
switch (condition) {
   case value0:
    statements
        break;
   case value1:
    statements
        break;
    def ault:
    statements
}
```

Ternary operator

```
// same as an if, but inline!
condition ? iftrue : iffalse;
```

Type conversion

`String` to any number	`int i = Intege r.p ars eInt(str);`
Any type to `String`	`String s = String.va lueOf(value) ;`
Between numeric types	`int i = (int) number;`

Array Utilities

Usage	Description
Arrays.copyOf(*og*, *og*.length);`	Copy array onto another one

Recursion

```
// method that sums all the number 1..10
// attention! recursion can incur in an infinite
loop
// handle it well
public static int sum(int start, int end) {
   if (end > start) {
        return end + sum(start, end - 1);
   } else {
        return end;
   }
}
```

Access Definitions

	public	protected	package	private
Class	•	•	•	•
Package	•	•	•	
Subclasses	•	•		
Classes	•			

Usage of an Object

```
// to use an object
public static void main (String[] args) {
       // declar ation meets constr uctor method
        Object obj = new Object (at tr1);
        obj.me thod();
}
```

Dynamic Data Types

```
import java.util.*;
L ist <type> names =   new ArrayL  is t <type>();
Vector<type> vec = new Vector< type>();
Stack<type> stk= new Stack< >();
```

Dynamic Data Types Methods

Vector	
`vec.add(value);`	Add an element to the end
`vec.add(i, value);`	Add an element to a specific index
`vec.ca pac ity();`	Returns the current capacity
`vec.cl ear();`	Empty vector
`vec.cl one();`	Clone the vector
Stack *Inherits from Vector*	
`stk.em pty();`	Returns true if head is empty
`stk.pe ek();`	Returns head of stack
`stk.pop();`	Delete head of stack
`stk.push(value);`	Add an element at head
`stk.se arch(value);`	Returns index of element