# Cheatsheet Java

## Comments

Single-line Comment:

```
String txt = "Hello!";
//this is a Comment
System.out.println(txt);

```

Multi-line Comment:

```
String txt = "Hello!";
/*Comments will not be
executed */
System.out.println(txt);
```

## Control structures

```
if(condition1){
  /*if condition1 true,
  execute*/
}
else if(condition2){
  /*if condition1 false and
  condition2 true, execute */
}
else{
  //if everything false, execute
}
```

## Loops

```
for(int i=0; i<10; i++){
  //execute 10 times
}
while(condition){
  //execute as long as condition
}
do{
  //execute at least once
}while(condition);
```

## Switch

```
switch(expression){
  case 1:
    //execute if expression==1
    break;
  case 2:
    //execute if expression==2
    break;
  default:
    /*execute if expression is
     not 1 or 2 */
    break;
}
```

## Types

Primitive data types:

| Type | Size | Type | Size |
|---|---|---|---|
| byte | 8 bit | float | 32 bit |
| short | 16 bit | double | 64 bit |
| int | 32 bit | Type | Value |
| long | 64 bit | char | 'a', 'G' |
| | | boolean | true, false |
| | | void | - |

Typecasting: $byte \rightarrow short \rightarrow char \rightarrow int \rightarrow long \rightarrow float \rightarrow double$

Non-Primitive data types:

| Type | Value |
|---|---|
| String | "Hello World!" |
| Array | int[] myNum = {10, 20, 30, 40}; |

## Declaration, Initialisation

Declaration: int a; String txt;

<Type>< Name>;

Initialisation: int b = 50; int b = a;

<Type><Name>=<Literal/Variable>;

Assignment: a = b; txt = "abc";

## Operations

Arithmetic:

| Operation | Example |
|---|---|
| + | 3 + 5 == 8 |
| - | 7 - 2 == 5 |
| * | 4 * 2 == 8 |
| / | 7 / 2 == 3 |
| % (Modulo) | 72 % 10 == 2 |

Comparison:

| Operator | Math | Example |
|---|---|---|
| > | $>$ | 5 > 2 |
| >= | $\geq$ | 5 >= 2 |
| < | $<$ | 10 < 21 |
| <= | $\leq$ | 5 <= 5 |
| == | $=$ | 5 == 5 |
| != | $\neq$ | -32 != 32 |

## Functions

```
//Delaration and Implementation
<ret-type> <func-name>(<para-type>
      <para-name>, ...){
    // function body
    //execute
    return <expression>;
}
//Function call
<func-name>(<argument>, ...);
```

## Arrays

```
//Declaration
<type>[] <name>;
int[] arr;
//allocation
<name> = new <type>[<size>];
arr = new int[5];
//or
<name> = {<element1>, ...};
arr = {1, 2, 3, 4, 5};
//Access
<name>[<index>];
arr[2] = 5;
```

## Strings

```
/*Strings are immutable and come
with a number of methods
already implemented*/
//Declaration
String <name>=new String(<value>);
String helloString=new String("
      hello");
//or
String <name>=<value>;
String helloString="hello";
//Small Selection of useful Methods
helloString.length();
helloString.charAt(<index>);
helloString.split(" ");
```

## Object-Oriented Programming

- Attributes:
  - define the state of an Object
  - Data
  - Describes the Object
  - Other names: fields, properties
- Methods:
  - describes behavior of an Object
  - Code/Function
  - Changes the state of the object
  - Or interacts with other objects

```
// Defining Class
class <class-name>{
    //Attributes
    <type> <var-name>;
    //Methods
    <ret-type> <func-name>(<para-
      type> <para-name>, ...){
        // function body
    }
}
class Room {
  int chairs = 4; //Attribute
  void addChairs(int chairs){
      this.chairs += chairs;
  } //Method
}
//Creating Object
<class-name> <obj-name> =
new <class-name>();
Room kitchen = new Room();
//Accessing Attributes and Methods
<obj-name>.<var-name>; //Attribute
kitchen.chairs;

<obj-name>.<func-name>
(<argument>, ...); //Method
kitchen.addChairs(2);
```

Official Documentation: https://docs.oracle.com/en/java/javase/18/docs/api/index.html
Educational: https://www.w3schools.com/java/