Cheatsheet Java

Comments

Single-line Comment:

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

```

Multi-line Comment:

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

Control structures

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

Loops

```java
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

```java
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;
}
```

Functions

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

Types

Primitive data types:

Type	Size
byte	8 bit
short	16 bit
int	32 bit
long	64 bit

Type	Size
float	32 bit
double	64 bit

Type	Value
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

Arrays

```java
//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

```java
/*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
  - Modifier always private, use Getter/Setter for access
- Methods:
  - describes behavior of an Object
  - Code/Function
  - Changes the state of the object
  - Or interacts with other objects
  - Modifier mostly public

```java
// Defining Class
class <class-name>{
  //Attributes
  <modifier> <type> <var-name>;
  //Methods
  <modifier> <ret-type> <func-name
      >(<para-type> <para-name>,
      ...){
    // function body
  }
}
class Room {
  private int chairs = 4; //
      Attribute
  public void addChairs(int chairs)
  {
    this.chairs += chairs;
  } //Method
}
```

```java
//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);

/*to access members of own class
     use keyword this:*/
this.<var-name>;
this.<func-name>(<argument>, ...);
this.chairs += 5;
```

Access modifiers to define access to an attribute or method:

- public: Anyone can access the member, default
- private: Only the class itself can access the member
- protected: Only the class itself and its subclasses can access the member

Constructor:

- same name as class
- will get called if a new object is created
- mostly used for Initialisation of attributes

```java
class <class-name> {
  public <class-name>(...){
    //constructor body
  }
  ...
}
class Student {
  public Student(String name, ...){
    this.name = name;
    ...
  }
}
```

Inheritance

```java
/* To give a subclass all members
of a superclass
to inherit use 'extends' keyword */
  class Vehicle {
    ...
  }
  class Car extends Vehicle {
    ...
  }
```

```java
/* use 'super' to refer
to the superclass */
class <Subclass-name> extends
<Superclass-name> {
  public <Subclass-name>(...){
    super();
  }

  /*use @Override to replace a
  method from the superclass */
  @Override
  public <Superclass-Method>(){
    /* calls the method
    of the superclass */
    super.<Superclass-Method>();
    //insert own code here
  }
}
```

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