Java

executive version JDK 11.0.4

General Information

public	... access privilege - access from everywhere
private	... access privilege - no access from outside
protected	... access privilege - access from package only
static vs nonstatic	
static	... execute when starting the class, always usable (without creating a class), can always be only one value
nonstatic	... requires the creation of a class, before access to it
public class Helloworld{	... head of class
public Hellowolrd(){...}	... constructor
}	
void	... return type(void,String,int[], ...)
System.out.print("h w")	... functionCall(argument)
javac Helloworld.java	... command for compile Helloworld.java → H....class
java Helloworld	... command for execute H....java with H....class

Numbers

```
byte total = 3 * 3;                  // 9        8 bit
short total = 3 * 3;                 // 9       16 bit
int total = 3 * 3;                   // 9       32 bit
int total = 5 + 2 * 3;               // 11
                                     // 3
long total = 3L;                     // 5.25    64 bit
float total = 1.50f + 3.75f;         // 5.25    32 bit
double total = 1.50 + 3.75;                     64 bit

int total = (int)5.4 + 3;            // 8 typecast
int nine = Integer.parseInt("9")     // 9
int modulo = 9 % 2                   // 1       rest
```

Operators

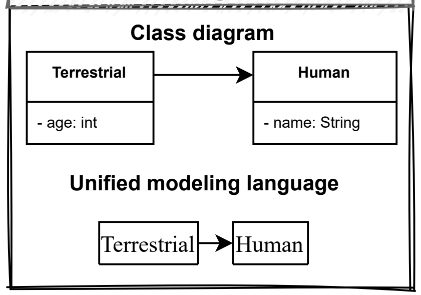
```
int a = 5 + 4;     // 9
int b = 5 - 4;     // 1
int c = 5 * 4;     // 20
int d = 5 / 4;     // 1
int e = 23 % 4     /* 3 modulo results in remainder
                   with integer division */
```

Strings

```
String name = "Pacman is yellow";            // Pacman is yellow              object
String.valueOf(name.charAt(0));              // P
name.length();                               // 6
name.split(" ");                             // ["Pacman","is","yellow"]
String.valueOf(total.split(" ")[0]);         // Pacman
String.join("Pacman", "is", "yellow")        // Pacmanisyellow
String nine = Integer.toString(9);           // "9"
total.replace("Pacman","Sun")                // "Sun is yellow"

char a = 'A';                                // A
char b = 'B';                                // B
System.out.println(a + b);                   // 131                           16 bit
String surname = "John";                     // John
String familyname = "Doe";                   // Doe
String fullname = surname + familiyname;     // JohnDoe = Concatenate / link Strings
```

UML / Class diagram



Class

```
public class Human {                                  // Human.java
   public String name;
   public Human (String name){
      this.name = name;
   }                                                  // heredity in UML
}
                                                      /* Terrestrial.java gets Human
public class Terrestrial extends Human {              attribute */
   private int age;                                   /* age is only available in this
   public Terrestrial(String name, int age){          class (private) */
      super(name);                                    // super(name) get attribut
      this.age = age;
   }

   public static void main(String[] args) {
      Terrestrial jdObject = new Terrestrial("John Doe",42);
      System.out.println(jdObject.name+" "+jdObject.age);
   }
}
         /*
         object vs class
         class:
           - blueprint, how does it work
           - won't be executed
           - can not do anything
         object:
           - concrete form of a class
           - accomplished functions
              (e.g. mercedes in my garage )
         */
```

Methods

```
public class MyClass {
  public static int methodName(double n){  // public ... see also @ General Information
    return (int) n;                        // static ... see also @ General Information
  }                                        // int ... return type, only one type possible
                                           /* double n ... transfer parameter as type double, more than one
                                           parameters possible */
                                           // return ... return value, not output value!

  public static String twoDigitsAtferComma(double number) {  // String method gets a number
    String str = String.format("%.2f", number);              // formats number into String
    return str;                                              // returns String
  }                                                          // method end

  public static void main(String args[]) {
    System.out.println(twoDigitsAtferComma(4.2/2) + " Euro");  /* "2.10 Euro", twoDigitsAfterComma
                                                               method call */
    System.out.println(methodName(5.2));                       // 5 , methodName method call
  }
}
```

Java *executive version JDK 11.0.4*

While Loops

```
int i = 0;                  // counter

while(true){                // As long as true
  i++;                      // i increment by 1
  if(i == 2){
     continue;              // go to next loop
  }
  if(i == 4){
     break;                 // end loop
  }
  System.out.print(i);  //  13
}
```

For Loops

```
int[] number = { 6, 5, 4, 3, 2, 1 };              // int Array

for (int i = 0; i < number.length; i++) {        /*  654321, access to
  System.out.print(number[i]);                   location of i in array
}                                                */

int[] number = { 1, 2, 3, 4, 5, 6 };
for (int i : number) {                           // for each
  System.out.print(i);                           //  123456
}
```

Lists

```
int[] numbers = {2,4,1};                 // int array → immutable
        index = 0   index = 1   index = 2   // index = location of number
double[] numbers;                        // double array
char[] letters = {'Y','E','S'};          // char array one-dimensional
int[][] matrix;                          // int array two-dimensional
int[][][][]  matrix_4x_4x;               // int array four-dimensional
int[] empty = new int[5];                // {0,0,0,0,0}
int[][] empty2 = new int[2][2];          // { {0,0} , {0,0} }
numbers.length                           // 3
```

Heredity

```
public class Terrestrial extends Human {}
```

```
public interface Polygon{} ... Polygon.java
public interface Circle{} ... Circle.java
public class Geometry implements Polygon, Circle{} ... Geometry.java
```

```
public abstract Animal{} ... Animal.java
public class Dog implements Animal{} ... Dog.java
```

/*
extends (normal heredity of classes) - only one super class
implements (heredity interfaces) - use as much interfaces as you need

interfaces: completely abstract, methods don't have a body
abstract classes: can be normal or abstract = methods that are always the same can get a body, there are no objects create-able, **see also @ Override**
*/

Imports / Java-packages

```
import java.util.Scanner;       // Scanner: used to intercept user input
import java.math.*;             // used for e.g. randomized numbers
  math.random(5);               // 0...5
  math.pow(5,2);                // 5^2 calculates potentiates
  math.round(5.9999);           //  6
import java.util.Arraylist;
  ArrayList<String> cars = new ArrayList<String>();
  cars.add("Volvo");
import java.lang.Exception;     //  see also @ Exceptions
import java.lang.Throwable;     //  see also @ Exceptions
```

Enum

```
public enum Color                       /* brief list
{                                       of various
  black, red, yellow, unknown;          acceptable
}                                       values */
```

Exceptions

```
try {                                               /* try ... catch ...
  int i = scanner.nextInt();                        try to do this (i) and
}
catch(InputMismatchException ime) {                 if error occurs handle this  with output
  System.out.println("Please enter an Integer");    "Please enter an Integer"
}
                                                    this program could go ahead*/

if(i<10){
  throw new IllegalArgumentException("wrong number");   /* throw error
}                                                       if i >= 10 output "wrong number"
                                                        this program stops immediately
                                                        */
```

Override

```
@Override                          /* overwrites methods of parent class e.g. to give the
                                   method a body in the interface */
public interface Polygon{
  public int acreage;
  public int perimeter;
}

                                   /*
public class Polygon{              public abstract class Polygon {
  @Override                          int a;
  public int acreage(){              int b;
    return a * b;                    public Polygon(int a, int b){
  };                                   public int acreage;
  public int perimeter(){              public int perimeter(){
    return 2 * (a + b);                  return a + b;
  }                                    }
}                                  */
```

Conditional Statements

```
boolean truth_value = true;   // true
                              // boolean ... express truth value true or false
double temp = 40;
if(temp <= 4){                                        // if(condition){do something}
   System.out.println("Winter is coming!");           // smaller than or equal to 4
} else if(temp > 4 && temp < 8){
   System.out.println("Fall is coming!");             // greater than 4 AND smaller than 8
} else if ((temp == 9) || (temp == 10)){              // = assignment, == boolean operator
   System.out.println("Fall is here!");               // 9 OR 10
} else if (!(temp < 11)){                             // NOT smaller than 11
   System.out.println("Climate change is here");      // "Climate change is here"
}

int weekday = 3;
String day;

switch (weekday) {                                    // switch(condition){
  case 1:                                             //    case condition:
    day = "Monday";                                   //      do something;
    break;                                            //      until break; ...}
  case 2:
    day = "Tuesday";
    break;
  default:
    day = "no saved day";                             // if condition is false do something
    break;                                            // break is necessary, otherwise loop
}
/*
!(x && y) is same as !x || !y
!(x || y) is same as !x && !y
!(a < 3 && b == 10) is same as a >= 3 || b != 10
*/

int a = 4;                                            // && = sequential conjunction
int b = 5;                                            //     || = sequential disjunction
if(a > 0 && b > 0){                                   /* sequential ... first I evaluate partial
}                                                     statements & then adjust the result */

int a = 4;                                            // & = strict conjunction
int b = 5;                                            //    | = strict disjunction
if(a > 0 & b > 0){                                    /* strict ... I evaluate statements from
}                                                     left to right and note partials */

int a = 4;
// int b = 5;
if(a > 0 & b > 0){   // int b == 'UNDEFINED';         // causes error
}
// true && true → works
// true & true → works
// true && UNDEFINED → works
// true & UNDEFINED → error
```