Exercise 1 Airplane
to be discussed in tutorials

An airplane model can be described (highly oversimplified) by the following attributes

- name: `String`
- empty weight: `double`
- number of seats: `int`
- fuel Consumption: `double`

a) Define a class `AirplaneModel` with the attributes defined above.

b) Augment your class with a default constructor and a constructor that initializes an airplane with only a name and number of seats. In addition to a constructor that initializes an airplane with a name, empty weight, number of seats, and fuel consumption.

c) Augment your class with the following methods:

- `getName()` to get the model's name
- `getEmptyWeight()` to get the model's empty weight
- `getSeats()` to get the number of seats
- `getFuelConsumption()` to get the fuel consumption
- `AddSeats(int x)`: to add `x` seats to an airplane.
- `display()` to display a full report about the airplane including name, empty weight, number of seats, and fuel consumption.
- `static void display(Airplane a)` to display a full report about the airplane including name, empty weight, number of seats, and fuel consumption.
- `compare(Airplane a)` to compare Airplane a with the Airplane on which the method will be invoked. The method returns the difference between the seat number of the Airplane on which the method is invoked and Airplane a.
- `static int compare(Airplane a, Airplane b)` to compare Airplane a with Airplane b. The method returns the difference between the seat number of Airplane a and Airplane b.

d) Augment your class with a method `main` that constructs one airplane model and performs the following

- Initialize one airplane model with the constructor that takes two parameters. The airplane has a name `Boeing` and `200` seats.
- Display the number of seats of the airplane.
- Add 50 seats to the airplane

- Display the number of seats of the airplane.
- Display a report about the airplane.
- Initialize another airplane model with the constructor that takes all parameters from the user.
- Compare the two Airplanes and print the Airplane name with the larger seat number. Try both the compare(Airplane a, Airplane b) and compare(Airplane a) methods.

Exercise 2 Complex Numbers

In this exercise, we will develop a complex number class.

A complex number needs two memory locations reserved for the real and imaginary parts and it must provide methods to perform operations such as addition and subtraction.

So the complex class will possess two floating point fields for the real and imaginary values and several methods to carry out the operations allowed on these values.

a) Define a class `Complex` with the attributes defined above.

b) Augment your class with a constructor that initializes the values.

c) Augment your class with the following methods:

- A method to get real and imaginary parts
- Two methods to add two complex numbers. The first method has only one parameter as input:

  ```
  public void add(Complex cvalue)
  ```

  The second method has two parameters as input (the method should create a new Complex object with the sum):

  ```
  public static Complex add(Complex cvalue1, Complex cvalue2)
  ```

- Two methods to subtract a complex number from another complex number.
- A method to display the real and imaginary parts in the following form

  ```
  4.1 + 3.9i
  ```

d) A `main` method to test your class.

Exercise 3 Time
To be discussed in the tutorial

Write a class named `Time` with the following attributes:

- Hours (`int`)
- Minutes (`int`)
- Seconds (`int`)

a) Define a class `Time` with the attributes defined above.

b) Augment your class with two constructors. The first constructor takes no arguments, and sets the hours, the minutes and the seconds to 0. The overloaded constructor takes the three arguments, hours, minutes and seconds.

c) Augment your class with the following methods:

- A method `hours()` that returns the hours.
- A method `minutes()` that returns the minutes.
- A method `seconds()` that returns the seconds.
- A method `addminute()` that will add one minute to the time.
- A method `addsecond()` that will add one second to the time.
- A method `addhour()` that will add one hour to the time.

Exercise 4 Politics
To be discussed in the labs

Develop a class `Country` with the following attributes:

- CountryName : `String`
- noOfCitizens : `int`
- isRoyal : `boolean`
- continent : `String`
- politicalState (4=peace, 3=increase intelligence, 2=Increase in force readiness,1=war): `int`

a) Define a class **Country** with the attributes defined above.

b) Augment your class with a constructor that intitalizes a country with a country name, number of citizens, whether it is a royal or not, the name of the continent and the politicalState. In addition to the default constructor.

c) Augment your class with the following methods:

- `getState()` to get the political state.
- `getRoyalState()` to get whether this country is royal.
- `setDefCon(int p)` to set the political state of the country.
- `increaseCitizen(int c)` to increase the number of citizens by a given value.
- `display()` to display a full report about the country.
- `compareTo(Country a)` to compare Country a with the Country on which the method will be invoked. The method returns the difference between the number of citizens of the Country on which the method is invoked and Country a.
- `static int compareTo(Country a, Country b)` to compare Country a with Country b. The method returns the difference between the the number of citizens of Country a and Country b.

d) Augment your class with a method `main` that constructs a country and performs the following

- Construct a country c1 and initialize it.
- Display a full report about the country.
- Set the political state of the country.
- Construct another country c2 and initialize it .
- Compare the two countries and print the country name with the larger number of citizens. Try both the compareTo(Country a, Country b) and compareTo(Country a) methods.

Exercise 5 Clerk

To be discussed in the tutorial

A clerk is given by the following attributes:

- his Name and lastname (of type `String`)
- the year of birth (of type `int`)
- his salary class (of type `int`)
- a boolean value that indicates whether the clerk is married.

The purpose of the assignment is to design a class for clerks and to find out his monthly salary given the information above. The monthly salary is calculated as follows:

The base salary of 2000 Euro will be increased by $x \cdot 17/9$ Percent, where x is the salary class. Starting with the year, in which the clerk will be 30 years old, the calculated amount increases each 5 years by 1 Percent The allowance for married clerks is 12.3 percent from the base salary.

a) Define a class `Clerk` with the attributes defined above.

b) Augment your class with a constructor `Clerk(String firstName, String lastName, int yearOfBirth, int salaryClass, boolean married)`. This constructor has to initialise the instance variables of a clerk with the actual parameters. In addition to the default constructor.

c) Augment your class with a method `static void promote(Clerk a, int newSalaryClass)` that will classify a clerk into a new salary class.

d) Augment your class with the methods `void marry()` and `void divorce()`, that change the marital status (married or not married).

e) Augment your class with a method `double salary(int thisYear)` that calculates the salary. In the variable `thisYear` the year, in which the salary has to be calculated, will be stored.

f) Augment your class with a method `static int compare(Clerk a, Clerk b)` to compare Clerk a with Clerk b. The method returns the difference between the salaries of a and b.

g) Augment your class with a method `main` that constructs two clerks and performs the following

- Display the salary of both clerks.
- The first clerk get married. Then display the salary of both clerks
- The second clerk change to the salary class 7.
- Compare the salaries of the two clerks and print the clerk name with a larger salary.