OOP
Lab 4.

🎯 Objectives

- Reading data from text/CSV file
- Arrays vs. ArrayList
- Implement **one-to-many** associations using ArrayList

Create a Project/Module with 3 packages:

- `lab4_1`
- `lab4_2`
- `lab4_3`

Each exercise has its own package and Main class (main method).

Exercise 1.

Structure:

- `lab4_1`
  - `Main`
  - `Person`

a. Add a text file named `lab4_1_input.txt` to the **project root folder**

b. Create a static method that reads the lines of a text file and prints them with line numbers to the standard output. Use the `hasNextLine()` and `nextLine()` methods of the `Scanner` class.

`lab4_1_input.txt` content:	**Standard output:**
apple peach plum one two three	1 apple 2 peach plum 3 one two three

```
public static void readFilePrintItsLineNumbered( String fileName ){
   try (Scanner scanner = new Scanner( new File(fileName))){
       //read and process the lines
   } catch (FileNotFoundException e) {
      e.printStackTrace();
```

```
    }
}
```

c. Test your method!

d. Add a CSV (Comma Separated Values) file named `lab4_1_input.csv` to the **project root folder**. For example:

```
John, BLACK ,  1980
Mary, WHITE ,        1982
Simon, Brown ,      2000
```

e. Study the ArrayList class:
- https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html
- https://www.w3schools.com/java/java_arraylist.asp

f. Create a class `Person` with 3 attributes: `firstName`, `lastName` and `birthYear`. Generate a `constructor` having 3 parameters, and generate `getter` methods for all the attributes. Generate a `setter` method for the lastName attribute. Generate a `toString` method.

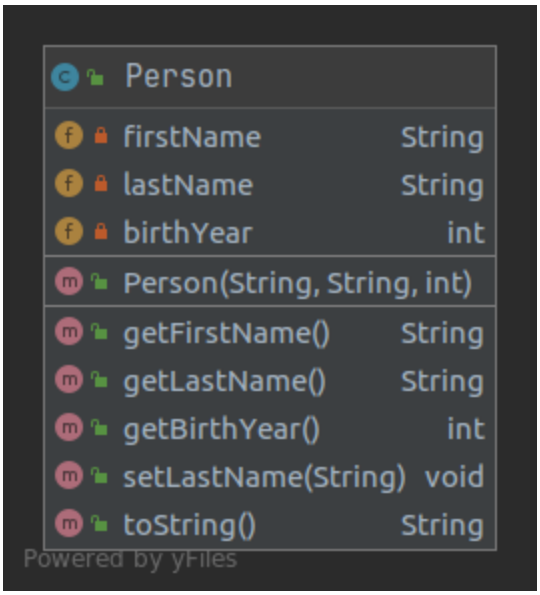

Fig. 1. Class Person

Create a static method that is responsible for reading the data from the CSV file, creating the `Person` objects, and storing them in an **`ArrayList`** (dynamic array).

```
public static ArrayList<Person> readFromCSVFile(String fileName) {
   ArrayList<Person> persons = new ArrayList<>();
   try (Scanner scanner = new Scanner(new File(fileName))) {
       while (scanner.hasNextLine()) {
           String line = scanner.nextLine();
           if (line.isEmpty()) {
               continue;
           }
           String[] items = line.split(",");
           // trim: eliminates leading and trailing spaces
           String firstName = items[0].trim();
           String lastName = items[1].trim();
           // Convert String → int: Integer.parseInt( String)
           int birthYear = Integer.parseInt(items[2].trim());
           persons.add(new Person(firstName, lastName, birthYear));
       }

   } catch (FileNotFoundException e) {
       e.printStackTrace();
   }
   return persons;
}
```

g. Test the previous function! Call `readFromCSVFile` (`main` method), then print the array to the standard output.

Exercise 2.

Structure:

- `lab4_2`
  - `Main`
  - `BankAccount`
  - `Customer`

a. Copy the `BankAccount` and `Customer` classes from `lab3_2`.

b. Modify the Customer class according to the class diagram (Fig. 2).

- Instead of array (`BankAccounts[] accounts`), use `ArrayList<BankAccount>`.

```
private ArrayList<BankAccount> accounts = new ArrayList<>();
```

- Delete `numAccounts,` and `MAX_ACCOUNTS`.

- Adapt the `addAccount`, `closeAccount`, `getAccount` and `toString` methods. Use the for-each loop

```
for(DataType item : array) {
//...
}
```

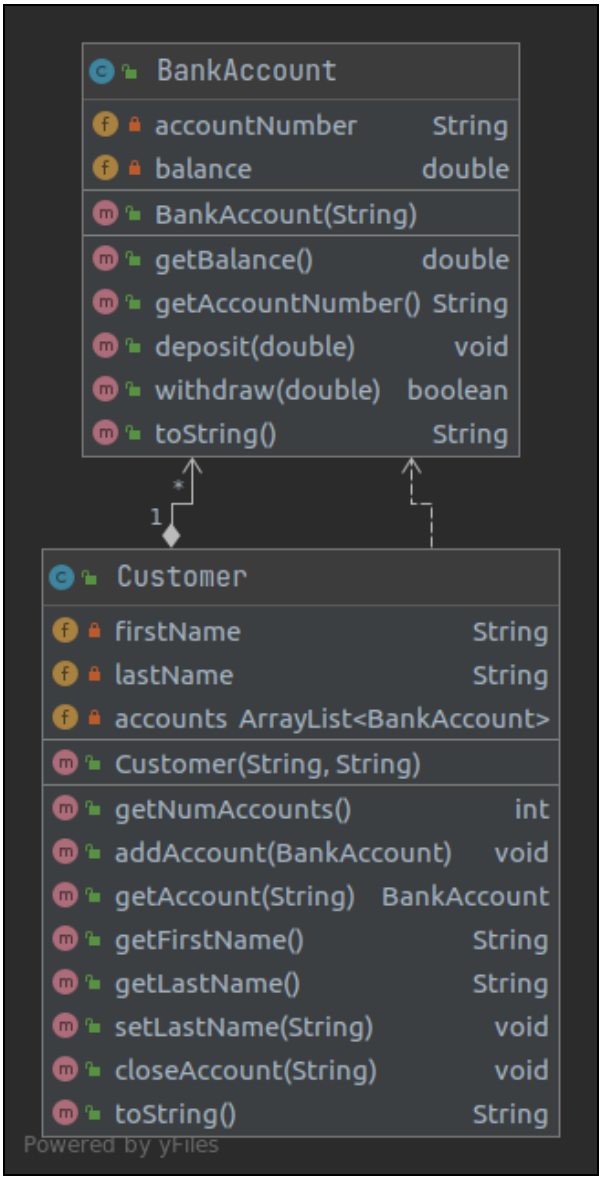

Fig.2. Class diagram: One-to-many relationship

c. Add a text file `lab4_2_input.csv` to the project root folder, which contains customers and their accounts data in the following format:

```
Customer, John, Black
Account, OTP1, 1000
```

```
Account, OTP3, 500
Customer, Steve, Brown
Customer, Mary, White
Account, OTP2, 1200
Account, OTP4, 300
Account, OTP5, 5000
```

The accounts of a customer are placed in the lines following the line containing customer's information.

d. `Main` class - `main` method:
   - Create an ArrayList of customers.
   - Fill the array with data read from the file.
   - Print the array to the standard output.

If you worked correctly, this is how the output will look like:

```
John Black accounts:
      BankAccount{accountNumber='OTP1', balance=1000.0}
      BankAccount{accountNumber='OTP3', balance=500.0}

Steve Brown accounts:

Mary White accounts:
      BankAccount{accountNumber='OTP2', balance=1200.0}
      BankAccount{accountNumber='OTP4', balance=300.0}
      BankAccount{accountNumber='OTP5', balance=5000.0}
```

Exercise 3.

Create a class `MyArray` that encapsulates an array of double values (see Fig.3). Your class should support the following constructors:

- Creation of a `MyArray` having a given size (length).
- Creation of a `MyArray` from a Java array `double[ ]` **(create a deep copy: allocation + copy of the array)**
- Creation of a `MyArray` from an existing `MyArray` **(create a deep copy).**
- Creation of a `MyArray` from a file that contains the size and the elements.

  Ex: `data.txt`

  ```
  5 // number of elements
  -1 -2 1 2 0 // the elements
  ```

and the following methods:

- `fillRandom` - fill the array with random numbers from [a,b)
- `mean` - return the mean of the array
- `stddev` - return the standard deviation of the array
- `sort` - sort the elements of the array
- `print` – print the elements of the array

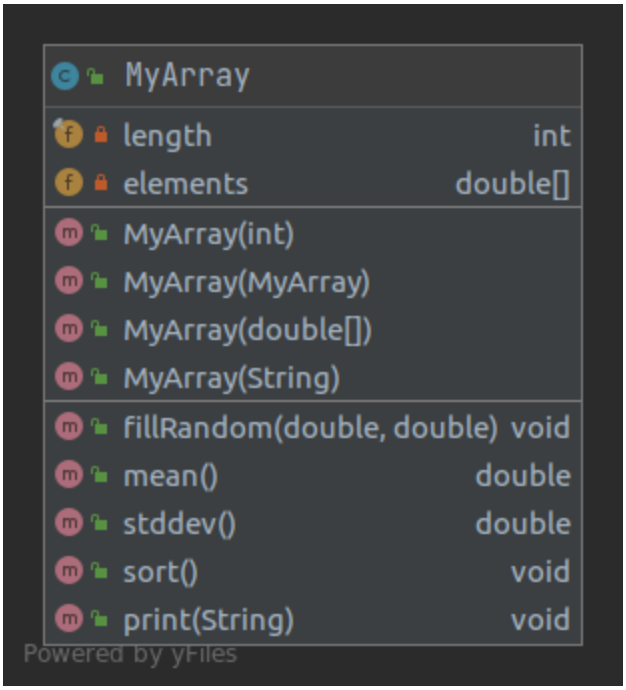

Fig. 3 Class MyArray

Test your class using the following main method:

```
public static void main(String[] args) {
   MyArray a1 = new MyArray(10);
   a1.print("a1");
   a1.fillRandom(0, 9);
   a1.print("a1");
   a1.sort();
   a1.print("a1");
```

```
   System.out.printf("\tMean: %10.2f Stddev: %10.2f\n",
a1.mean(), a1.stddev());
   double t[] = {4, 9, 0, -34, 28, 76, 100, -1};
   MyArray a2 = new MyArray(t);
   a2.print("a2");
   a2.sort();
   a2.print("a2");
   System.out.printf("\tMean: %10.2f Stddev: %10.2f\n",
a2.mean(), a2.stddev());
   MyArray a3 = new MyArray("lab4_3_input.txt");
   a3.print("a3");

   MyArray a4 = new MyArray( a3 );
   a3.sort();
   a3.print("a3");
   System.out.printf("\tMean: %10.2f Stddev: %10.2f\n",
a3.mean(), a3.stddev());
   a4.print("a4");
}
```

If you worked correctly, this is how the program output ends:

```
a2:   4.00   9.00   0.00 -34.00  28.00  76.00 100.00  -1.00
a2: -34.00  -1.00   0.00   4.00   9.00  28.00  76.00 100.00
      Mean:      22.75 Stddev:      41.34
a3:  -1.00  -2.00   1.00   2.00   0.00
a3:  -2.00  -1.00   0.00   1.00   2.00
      Mean:       0.00 Stddev:       1.41
a4:  -1.00  -2.00   1.00   2.00   0.00
```