

Objectives

- Using ArrayList
- Reading from CSV files
- Writing to CSV files
- Using *one-to-one* and *one-to-many* relationships

Structure:

- `oop.labor05` (package)
 - `model` (package)
 - `Course.java`
 - `Student.java`
 - `Training.java`
 - `Main.java`

We are an IT Training company offering Java courses to our customers. We have several **courses** stored in the `courses.csv` file. Each **course** has a name (no spaces are allowed), a short description, and a duration in hours. We regularly run **training** sessions in which we teach the curriculum of the courses. We teach a single course in each training session, which also has a start and an end date, and a price as well. **Students** may enroll in the training. As we teach online, the number of students is not limited.

Your task is to write a Java program which manages our training sessions!

1. Course class

Implement the Course class shown in Fig. 1. Note that the instances of the class have to be **immutable**!

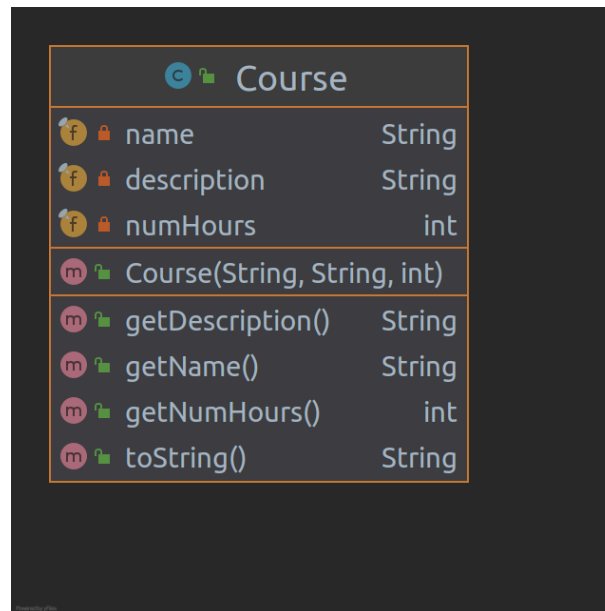


Fig. 1 Course class

Test the Course class!

- Add the `courses.csv` file to the module project folder (labor05)!
- Create a `readCourses` method in the `Main` class and read the contents of the `courses.csv` file and place it in an `ArrayList` (see Lab4).

```
private static ArrayList<Course> readCourses(String filename){//...}
```

- Call the `readCourses` in the main method, then print the courses to the standard output.

Before running your program, change the current working directory. This ensures IntelliJ can find your input files.

Edit configuration -- Working directory

.../IdeaProjects/2023.Oop/labor05

2. Student class

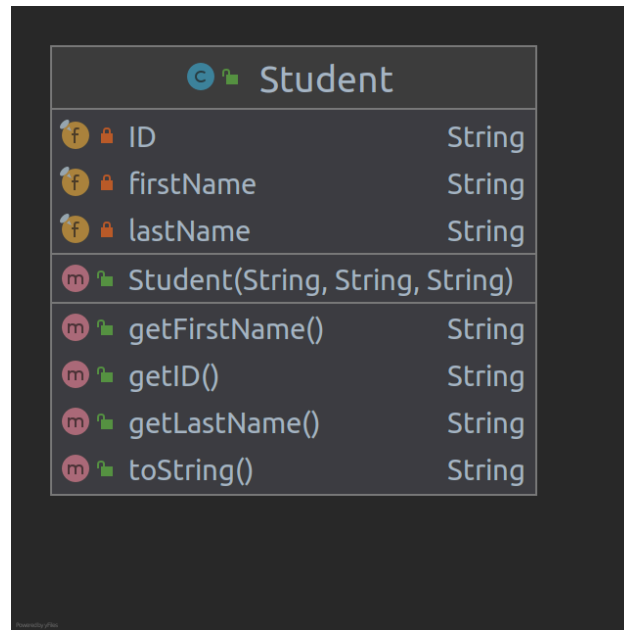


Fig. 2 Student class

Implement the Student class shown in Fig. 2. Note that the instances of the class have to be **immutable**!

Test the Student class!

- Add the `students.csv` file to the module project folder (labor-05)!
- Create a `readStudents` method in the `Main` class and read the contents of the `students.csv` file and place it in an `ArrayList`.

```
private static ArrayList<Student> readStudents(String filename){//...}
```

- Call the `readStudents` in the `main` method, then print the students to the standard output.

3. Training class

Each **Training** instance has a single **Course** (*one-to-one* association, $1 \rightarrow 1$).

Each **Training** instance may have several **Students** enrolled (*one-to-many* association, $1 \rightarrow *$).

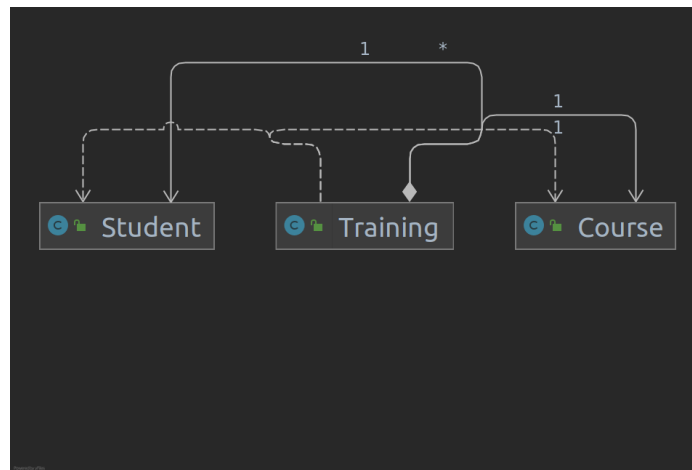


Fig.3 Package model

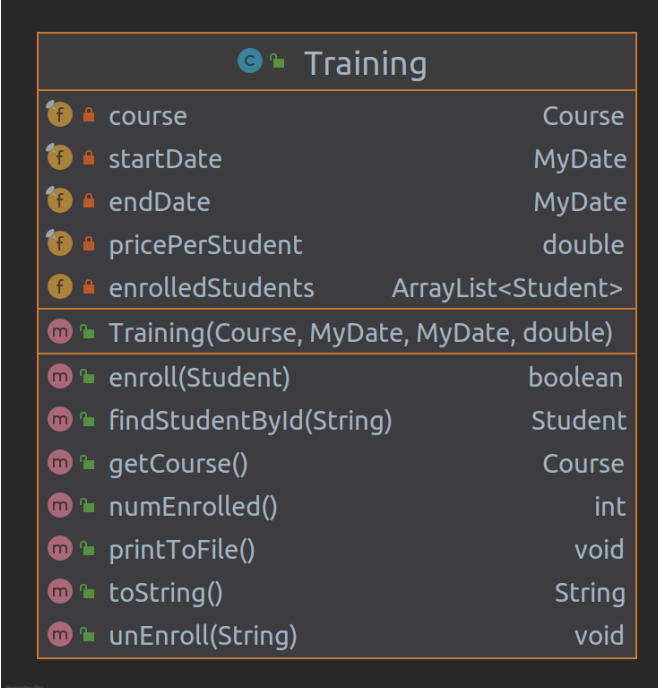
- Implement the `Training` class shown in Fig.4. Use the `MyDate` class (LAB2) for the start and end date of the training.
- Also note that each student can only enroll once for training (`enroll` method)!
- The `printToFile` method writes the enrolled students into an output file. The name of the file depends on the training data: `courseName_startdate_enddate.csv`

For example, for a JavaSE17 course between March 21-25, 2022 the filename should be: `JavaSE17_2022.3.21_2022.3.25.csv`. Use [String.format](#).

- The `toString` method prints the training session data as follows:

```
Training{
  course='Course{name='DPJava', description='Design Patterns in Java', numHours=24}'
  startDate=year: 2022 month: 3 day: 21
  endDate=year: 2022 month: 3 day: 25
  Student{ID='2871119070718', firstName='Laszlo', lastName='Franciska'}
```

```
Student{ID='1800923051246', firstName='Juhasz', lastName='Jacint'}  
Student{ID='1720626020015', firstName='Biro', lastName='Tamas'}  
Student{ID='1780824043217', firstName='Godri', lastName='Istvan'}  
Student{ID='1850917060518', firstName='Bodo', lastName='Elod'}  
Student{ID='2901121070018', firstName='Bogacs', lastName='Katalin'}  
Student{ID='1861018060618', firstName='Bege', lastName='Istvan'}  
Student{ID='2891220070818', firstName='Berekmeri', lastName='Emoke'}  
Student{ID='2820614052018', firstName='Kovacs', lastName='Emese'}  
Student{ID='2810513041018', firstName='Zsido', lastName='Maria'}  
}
```



Training	
course	Course
startDate	MyDate
endDate	MyDate
pricePerStudent	double
enrolledStudents	ArrayList<Student>
Training(Course, MyDate, MyDate, double)	
enroll(Student)	boolean
findStudentById(String)	Student
getCourse()	Course
numEnrolled()	int
printToFile()	void
toString()	String
unEnroll(String)	void

Fig. 4 Training class

Test the Training class!

Main class - main method:

- Create trainings and enroll students in them
 - For each course create a separate Training having the following features:
 - The price should be generated randomly between 1000 and 2000 EUR.
 - The startDate will be 2023.03.21 and the end date 2023.03.25.
 - Enroll 10 students randomly: generate a random number between 0 and size of the student list. Enroll the student from the random position to the training. Note

that the same position may be generated more than once. You should have exactly 10 students enrolled in each training.

- For each student list the enrolled training sessions in the following format:

Godri Istvan

JavaSE8, Java Standard Edition 8

JavaSpringBoot2, Developing REST Microservices

DPJava, Design Patterns in Java