Java Persistence API

Cascades Workshop

In this workshop we will practice the cascades.
First we use the persist cascading
And then we look at the remove cascading

Persist cascade

This is the code that we need in our test class:
Create 3 students and one tutor and persist them all.

```
Tutor t1 = new Tutor("ABC123", "Teacher 1", 290000);
em.persist(t1);

Student s1 = new Student("Student1", "1-STU-2019");
t1.addStudentToTeachingGroup(s1);

Student s2 = new Student("Student2", "2-STU-2018");
t1.addStudentToTeachingGroup(s2);

Student s3 = new Student("Student3", "3-STU-2017");
t1.addStudentToTeachingGroup(s3);

em.persist(s1);
em.persist(s2);
em.persist(s3);
```

Change the **`'update'` to `'create'`** in *persistence.xml*

Run the code. We have now one teacher and three students.
What we have done here is that we have manually called *persist()* on every object we created.
What if we forget one of them? What happens in the database?

There is a setting that can automatically call the persist method for us. For example when we persist a teacher, it would automatically persist the unpersisted students.

Persisting tutor and automatically students will be persisted

In Tutor class above the set of students add
`cascade=CascadeType.PERSIST` in the @oneToMany annotation:

```
@OneToMany(cascade = CascadeType.PERSIST)
private Set<Student>teachingGroup;
```

And the import: `jakarta.persistence.CascadeType`

Now remove the persist method on the three students in the main method.
Run the code. It should work. Check the student table if there are three students there.

A method that can create students and add them to a tutor's teaching group.

Another thing that we can do and make it more automatic, is to write a method that can create students and add them to the tutor's teaching group.

Now we write a method in Tutor class that does two things:

- creating students
- adding the students to the group:

```
public void createStudentAndAddtoTeachingGroup(String studentName, String
                    enrollmentID) {
      Student student = new Student(studentName, enrollmentID);
      this.addStudentToTeachingGroup(student);
}
```

So we do not need to create the students and add them to the group in the main method manually.

So remove the code that creates, persists and adds students to the group and instead add this code:

```
t1.createStudentAndAddtoTeachingGroup("Student1", "1-STU-2019");
t1.createStudentAndAddtoTeachingGroup("Student2", "2-STU-2018");
t1.createStudentAndAddtoTeachingGroup("Student3", "3-STU-2017");
```

Run the code and check if you got all the students in the table.
It works because we have set the cascade option.

Cascade remove

We can add this CascadeType in our Tutor class:

```
@OneToMany(cascade = {CascadeType.PERSIST, CascadeType.REMOVE})
private Set<Student>teachingGroup;
```

Now if we want to remove a tutor, hibernate removes all the students that are related to this tutor.

Change create to update in persistence.xml file.
In the main method comment out the created tutor and students.

Add the following instead:

```
Tutor t1 = em.find(Tutor.class, 1);
em.remove(t1);
```

Run the code and check the result. If you open the *tutor* table, there are no tutors in it. And there is no longer a student in the student table either.

When using *cascade remove* you should be very careful, because as you saw now it deletes everything related to the object that we want to delete and maybe it is not what you want in your project.

We can remove *CascadeType.REMOVE* from the Tutor class now.