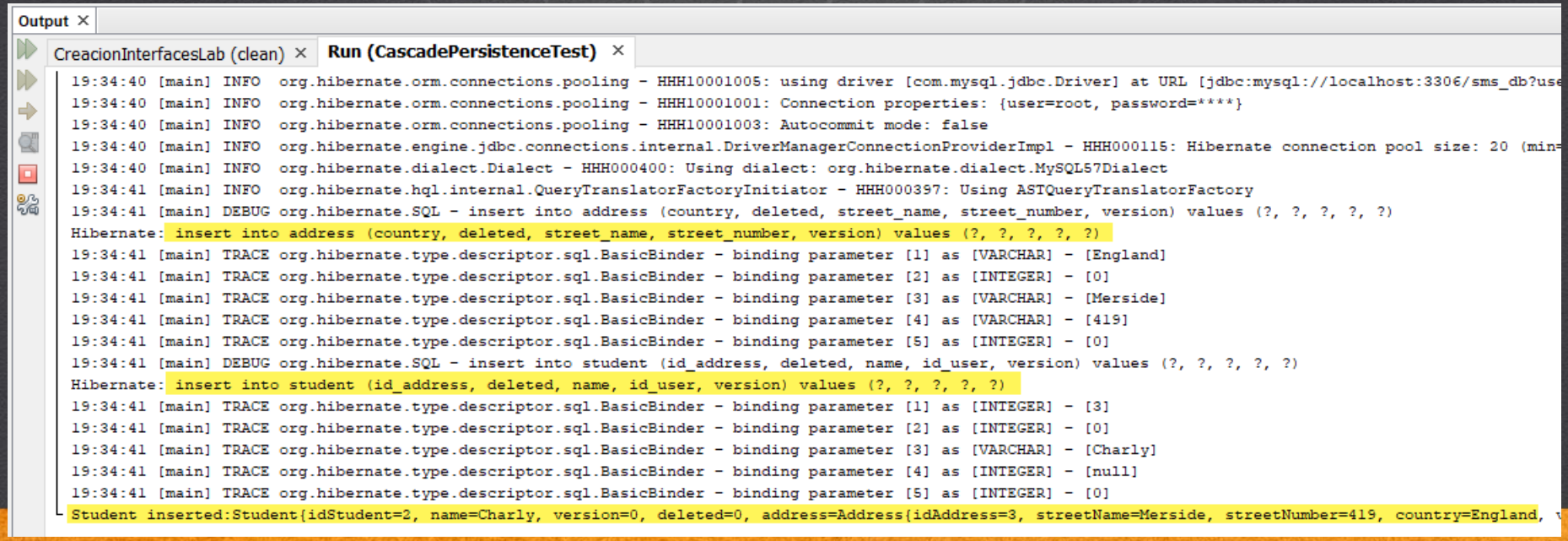
HIBERNATE & JPA COURSE

EXERCISE

CASCADE PERSISTENCE WITH HIBERNATE/JPA

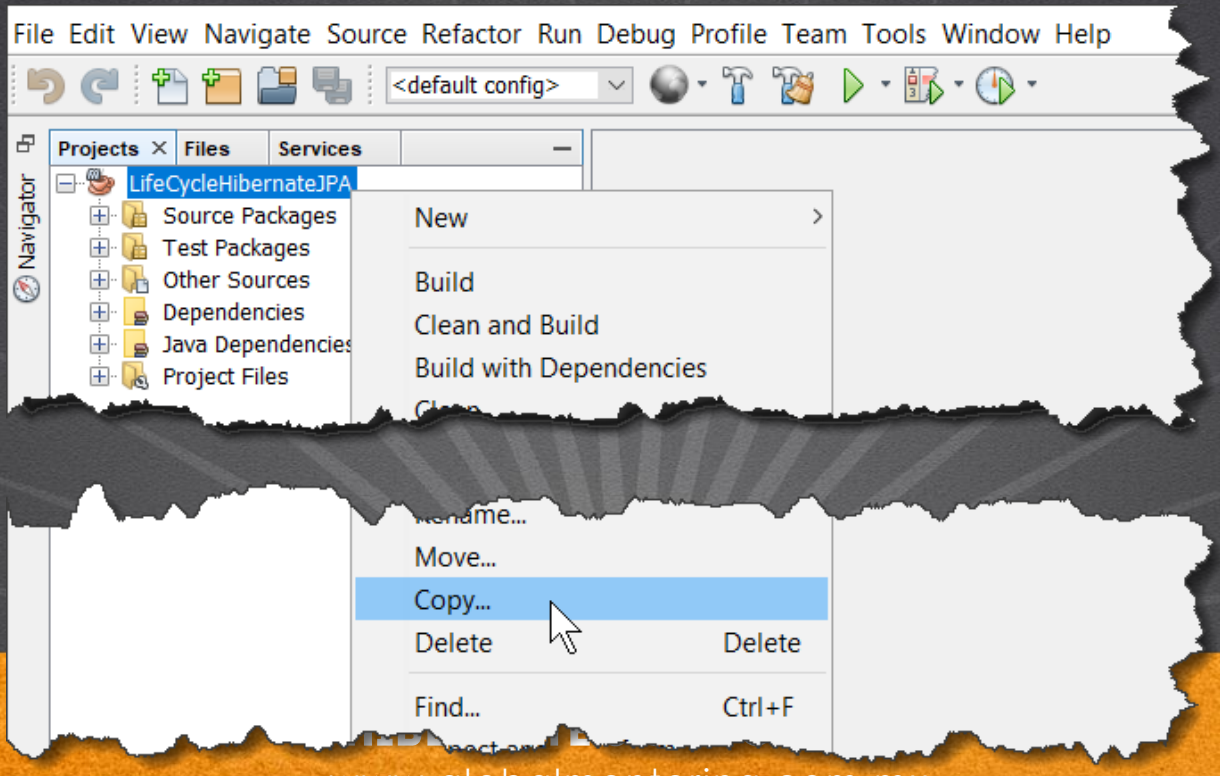
EXERCISE OBJECTIVE

Create a project to implement cascade persistence using Hibernate and JPA. At the end we should observe the following:



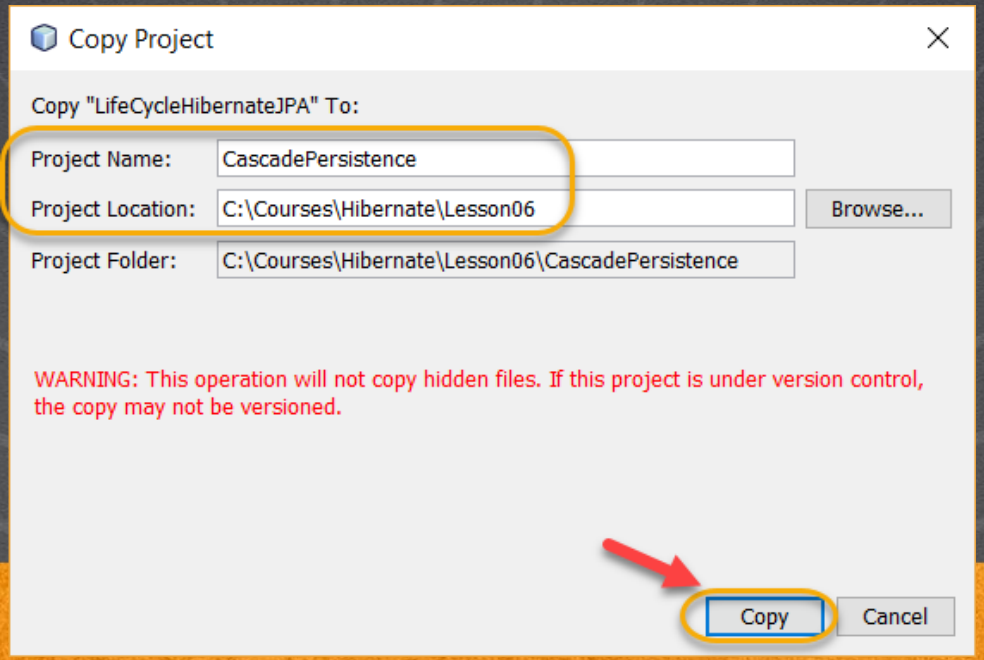
1. CREATE A PROJECT

We copy and paste the previous project:



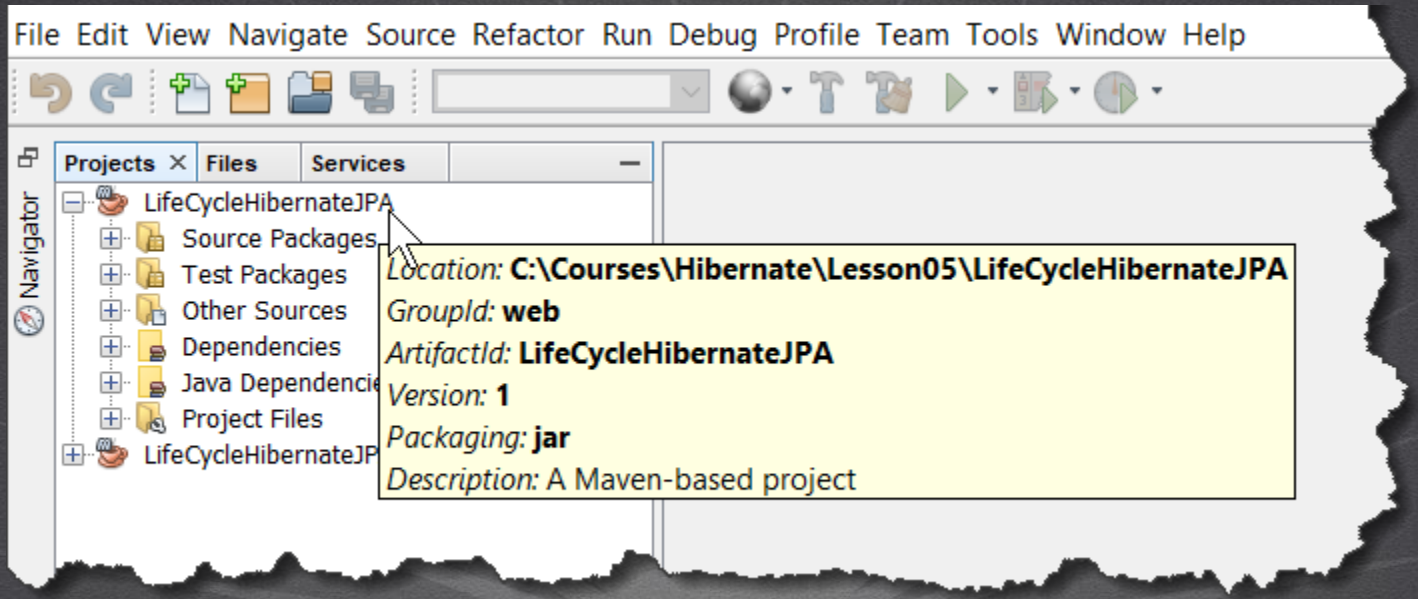
1. CREATE A PROJECT

We copy and paste the previous project, and we rename it to CascadePersistence:



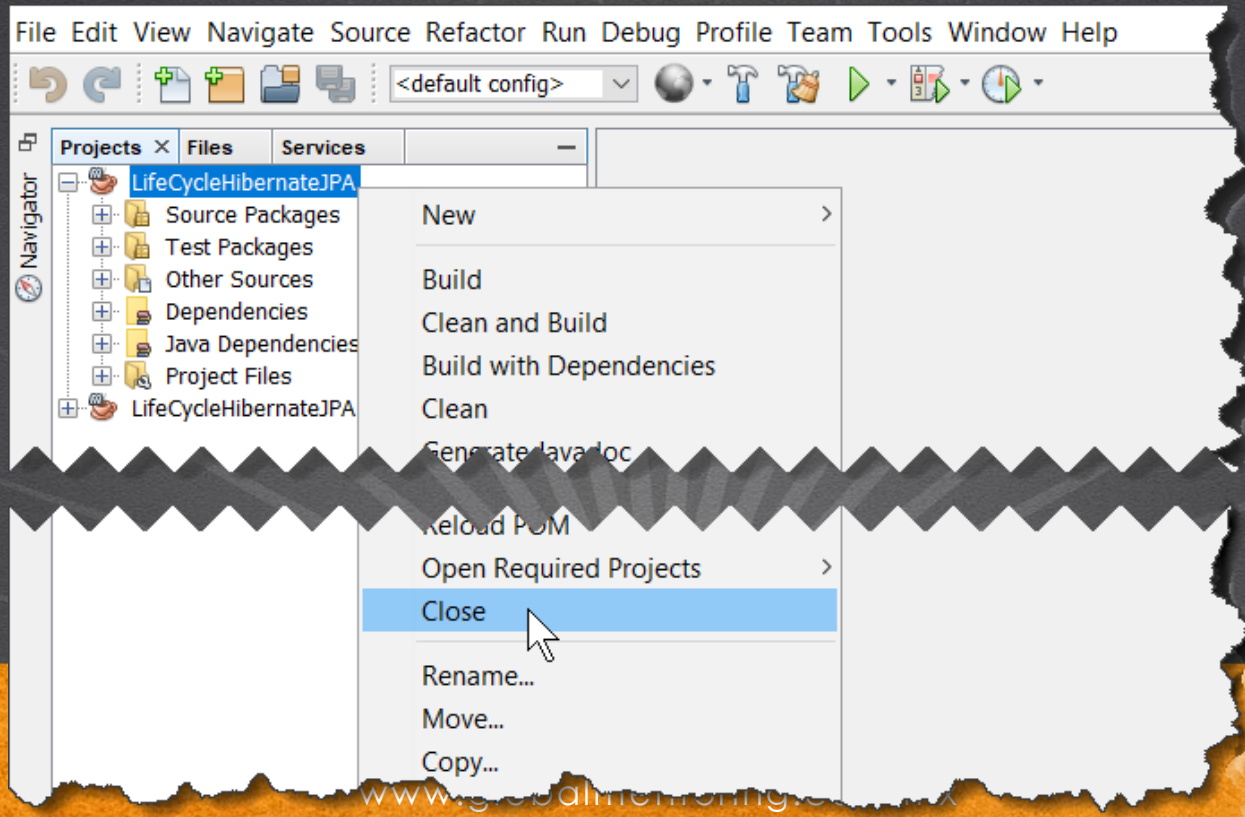
2. CLOSED THE PREVIOUS PROJECT

We locate the project that we are going to close:



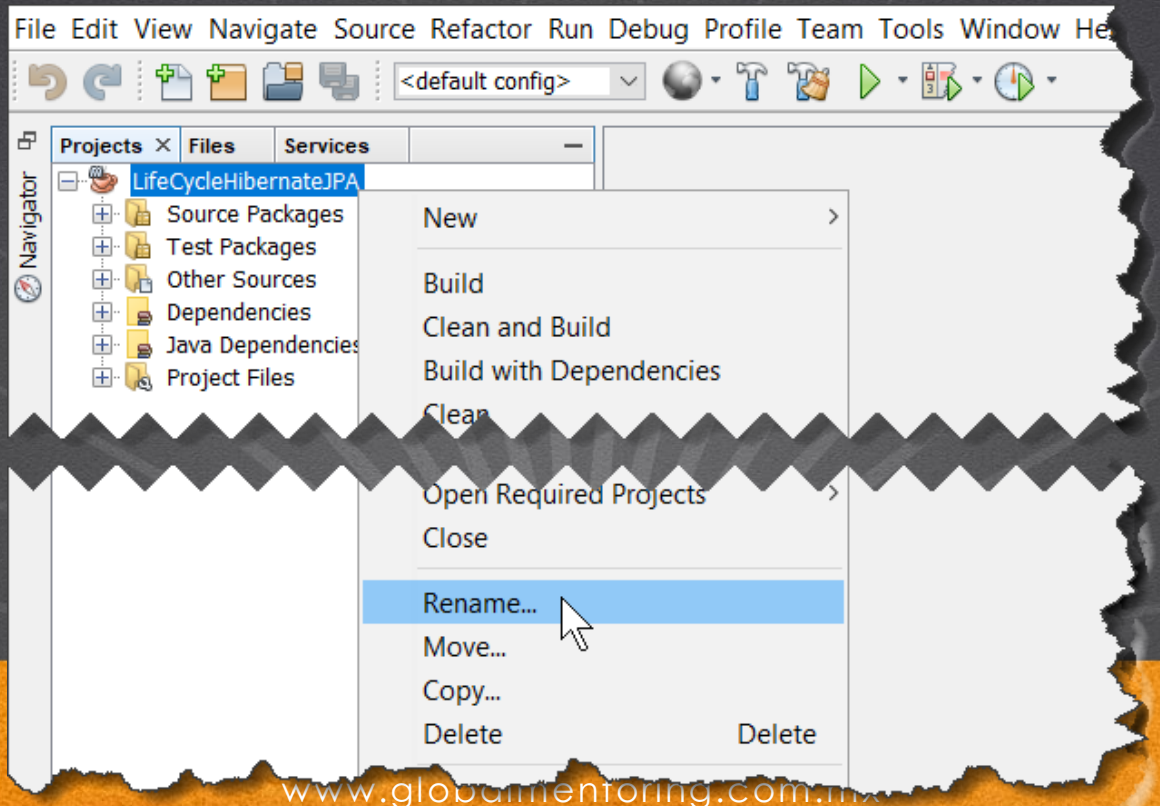
2. CLOSED THE PREVIOUS PROJECT

We close the project that we will no longer use:



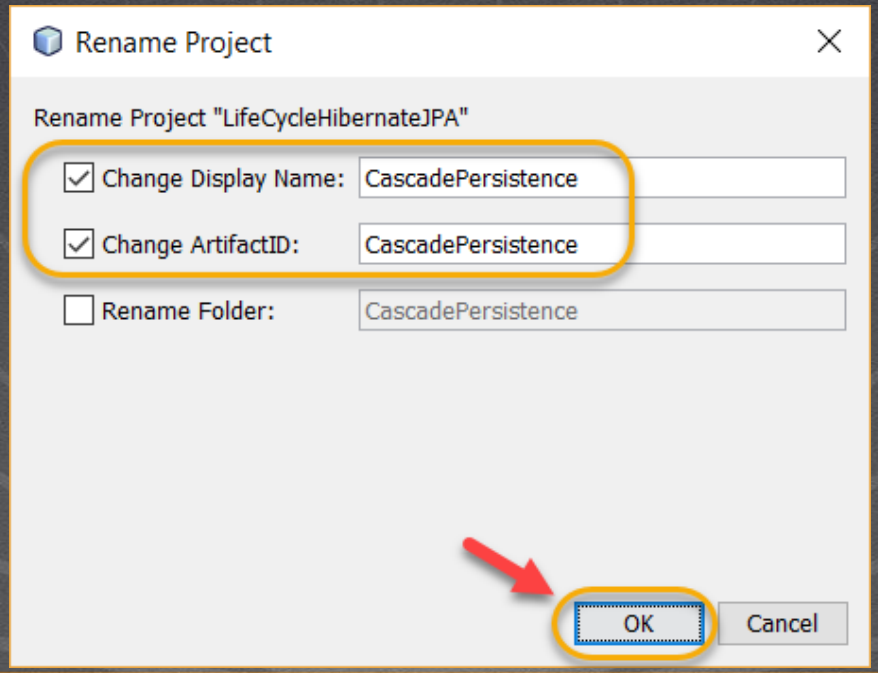
3. RENAME THE PROJECT

Rename the Project:



3. RENAME THE PROJECT

Rename the Project:



4. MODIFY A CLASS

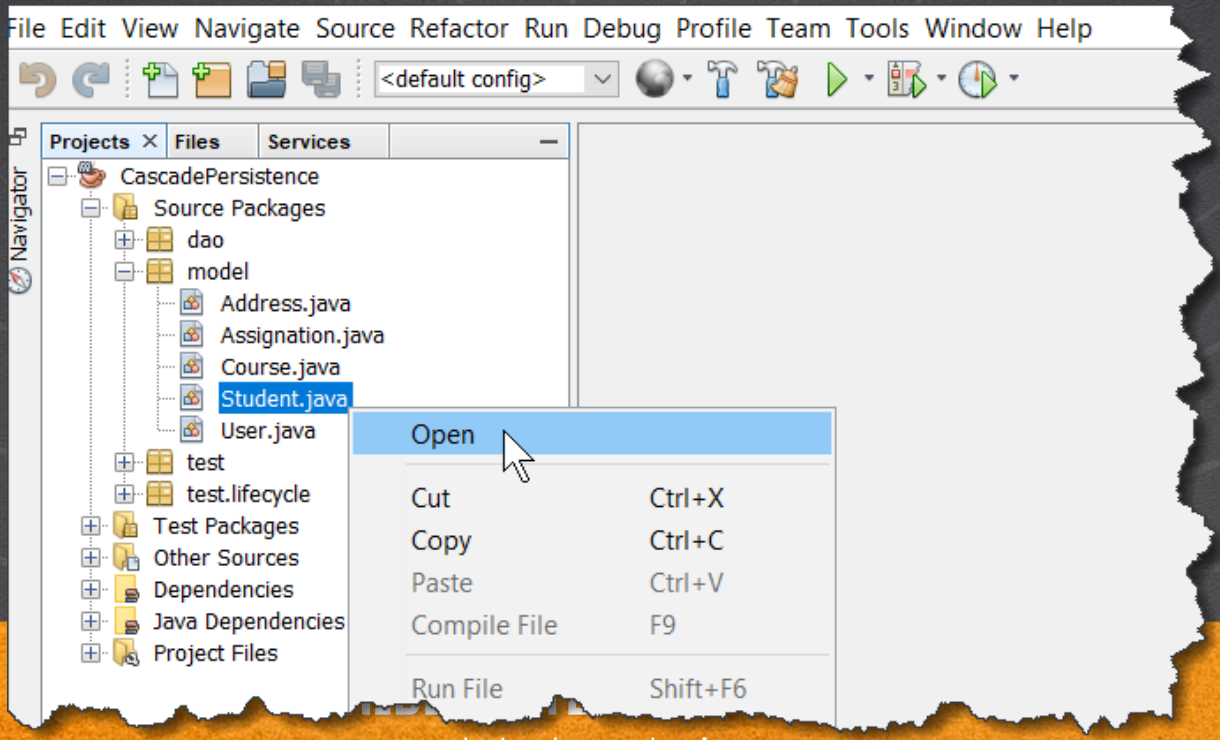
Modify the model.Student.java class to add the concept of cascade persistence in the address attribute.

In other words, when saving a Student object, it will also save the related address object automatically. We will use the following annotation:
@ManyToOne(cascade = CascadeType.ALL)

Let's see how our student class is.

4. MODIFY A CLASS

Modify the Student.java class:



4. MODIFY THE CODE

Student.java:

Click to download

```
package model;

import java.io.Serializable;
import java.util.*;
import javax.persistence.*;

@Entity
@Table(name = "student")
@NamedQueries({
    @NamedQuery(name = "Student.findAll", query = "SELECT s FROM Student s")})
public class Student implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_student")
    private Integer idStudent;

    private String name;

    private int version = 0;

    private int deleted = 0;

    @OneToMany(mappedBy = "student")
    private List<Assignation> assignationList;
```

4. MODIFY THE CODE

Student.java:

Click to download

```
@JoinColumn(name = "id_address", referencedColumnName = "id_address")
@ManyToOne(cascade = CascadeType.ALL)
private Address address;

@JoinColumn(name = "id_user", referencedColumnName = "id_user")
@ManyToOne
private User user;

public Student() {
}

public Student(Integer idStudent) {
    this.idStudent = idStudent;
}

public Student(Integer idStudent, String name, int version, int deleted) {
    this.idStudent = idStudent;
    this.name = name;
    this.version = version;
    this.deleted = deleted;
}

public Integer getIdStudent() {
    return idStudent;
}

public void setIdStudent(Integer idStudent) {
    this.idStudent = idStudent;
}
```

4. MODIFY THE CODE

Student.java:

Click to download

```
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}


public int getVersion() {
    return version;
}

public void setVersion(int version) {
    this.version = version;
}

public int getDeleted() {
    return deleted;
}

public void setDeleted(int deleted) {
    this.deleted = deleted;
}

public List<Assignation> getAssignationList() {
    return assignationList;
}
```

4. MODIFY THE CODE

Student.java:

Click to download

```
public void setAssignationList(List<Assignation> assignationList) {
    this.assignationList = assignationList;
}

public Address getAddress() {
    return address;
}

public void setAddress(Address address) {
    this.address = address;
}

public User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (idStudent != null ? idStudent.hashCode() : 0);
    return hash;
}
```

4. MODIFY THE CODE

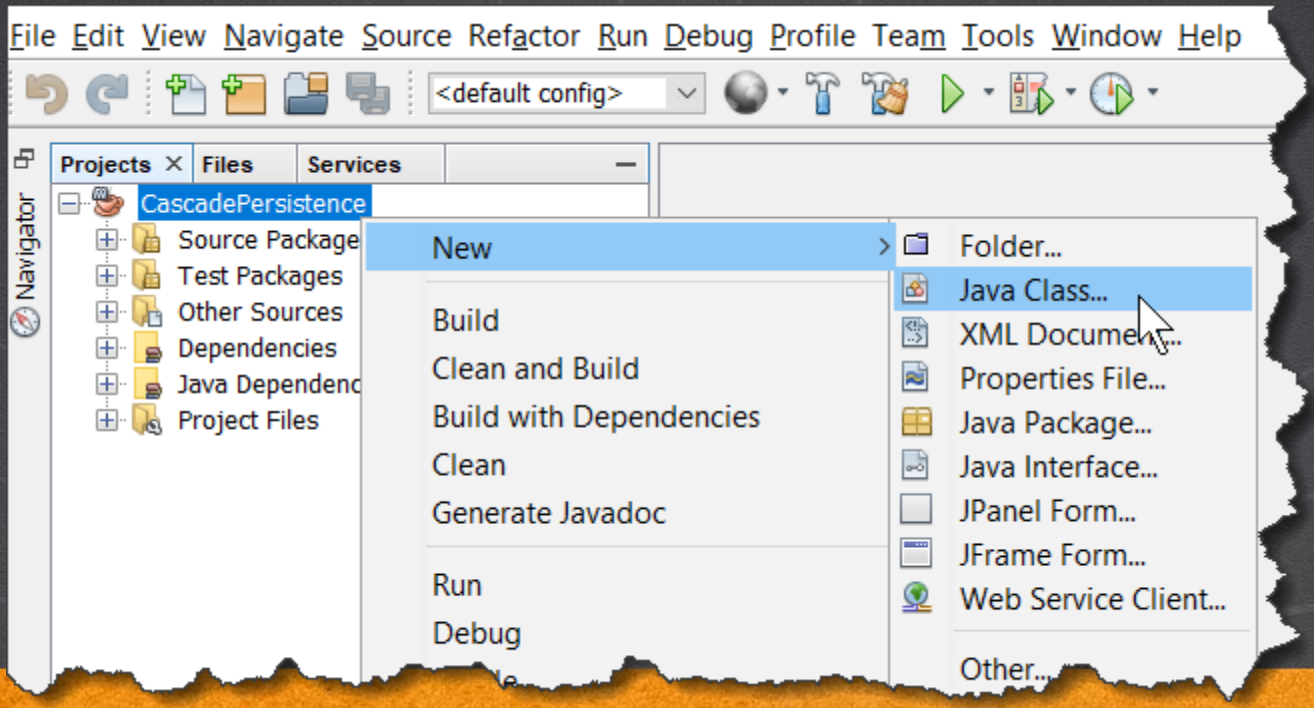
Student.java:

Click to download

```
    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Student)) {
            return false;
        }
        Student other = (Student) object;
        if ((this.idStudent == null && other.idStudent != null) || (this.idStudent != null &&
!this.idStudent.equals(other.idStudent))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Student{" + "idStudent=" + idStudent + ", name=" + name + ", version=" + version + ", deleted=" + deleted +
", address=" + address + ", user=" + user + '}';
    }

}
```

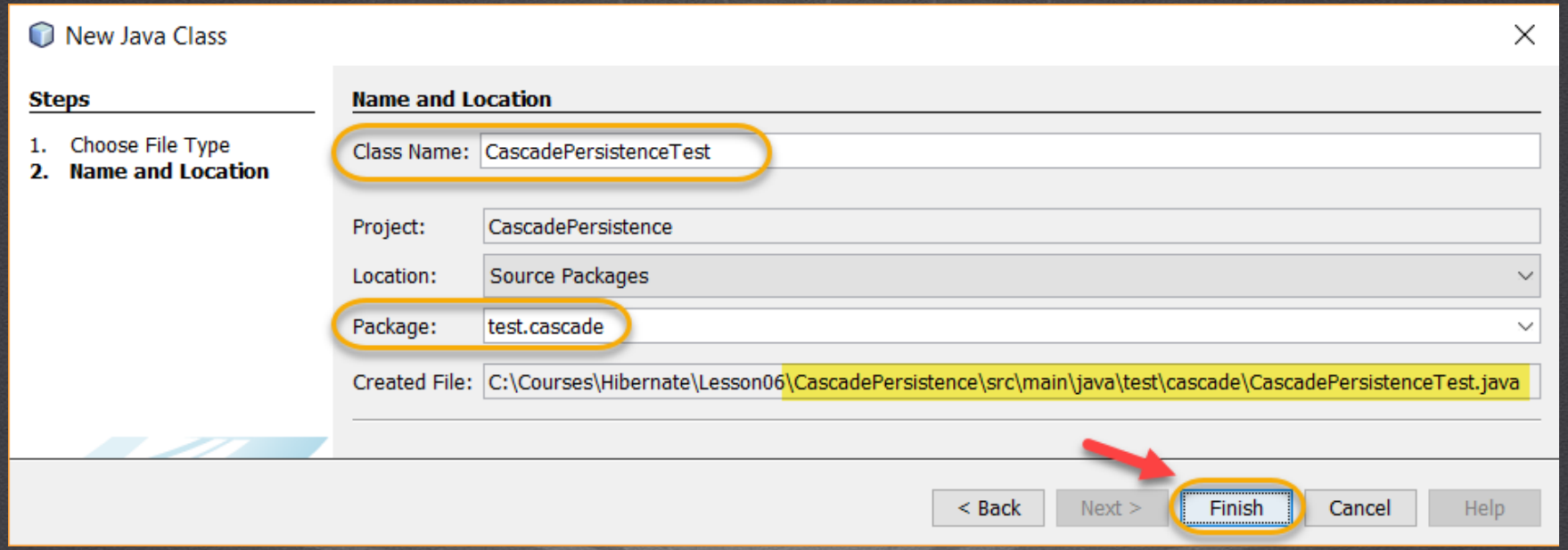
5. CREATE A NEW CLASS

We create a CascadePersistenceTest.java class:



5. CREATE A NEW CLASS

We create a CascadePersistenceTest.java class:



6. MODIFY THE CODE

CascadePersistenceTest.java

Click to download

```
package test.cascade;

import javax.persistence.*;
import model.*;

public class CascadePersistenceTest {

    public static void main(String[] args) {
        EntityManagerFactory factory = Persistence.createEntityManagerFactory("HibernateJpaPU");
        EntityManager em = factory.createEntityManager();

        // We create an Address object
        Address address = new Address();
        address.setStreetName("Merside");
        address.setStreetNumber("419");
        address.setCountry("England");

        // We create a Student object
        Student student = new Student();
        student.setName("Charly");

        //We add the relationship and its persistence in cascade
        student.setAddress(address);
```

6. MODIFY THE CODE

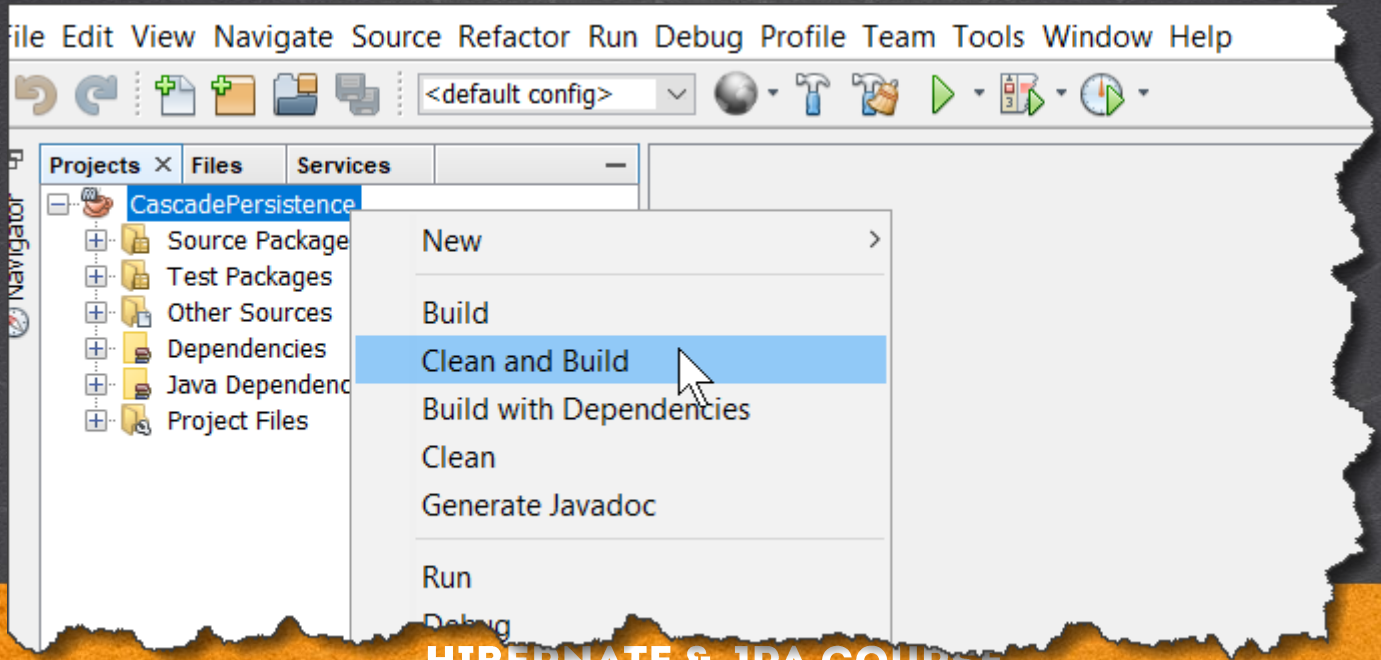
CascadePersistenceTest.java

Click to download

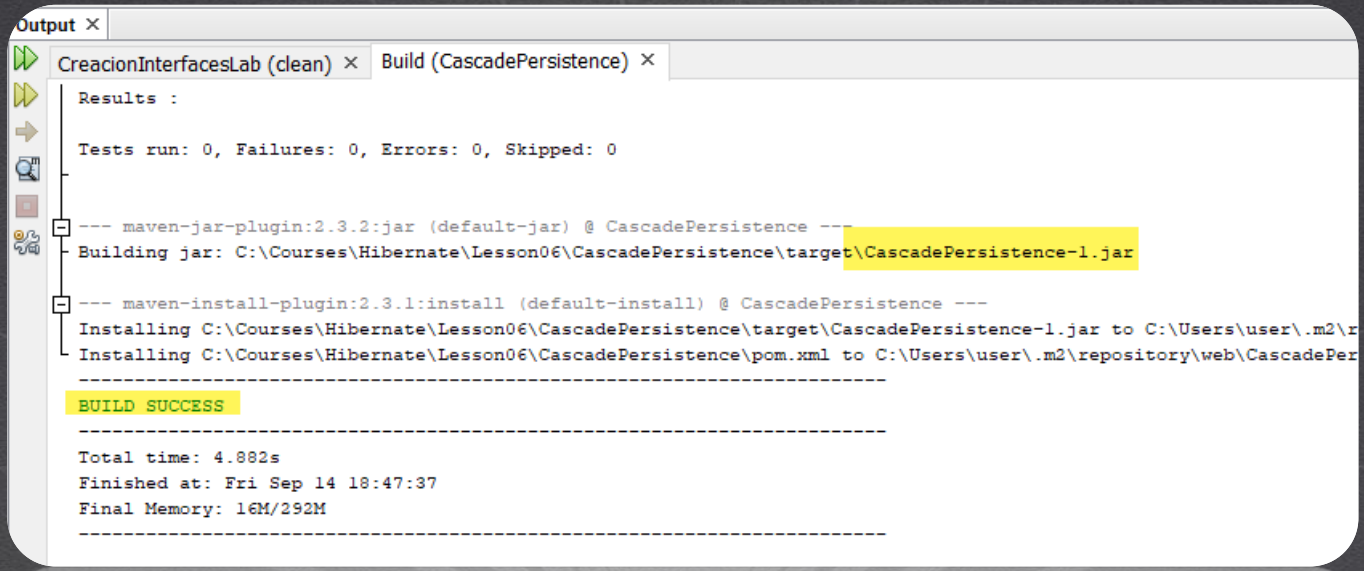
```
        try {
            em.getTransaction().begin();
            //We only persist the student, and the associated
            //relationships marked as cascading persistence are added automatically
            em.persist(student);
            em.getTransaction().commit();
        } catch (Exception e) {
            em.getTransaction().rollback();
            e.printStackTrace(System.out);
        } finally {
            if (em != null) {
                em.close();
            }
        }

        // Embedded objects
        System.out.println("Student inserted:" + student);
    }
}
```

8. EXECUTE CLEAN & BUILD

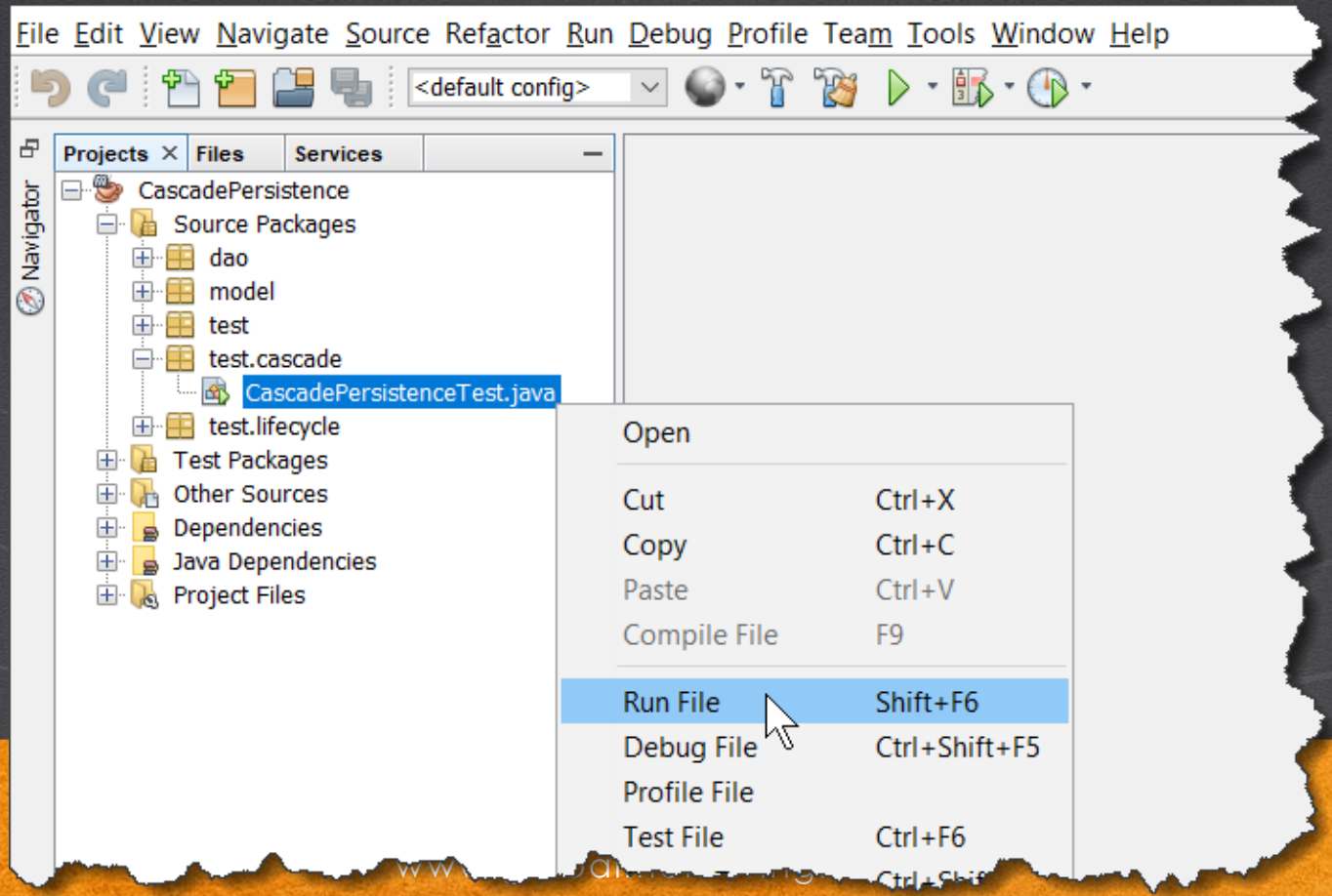
We do Clean & Build to have the latest versions of each file:



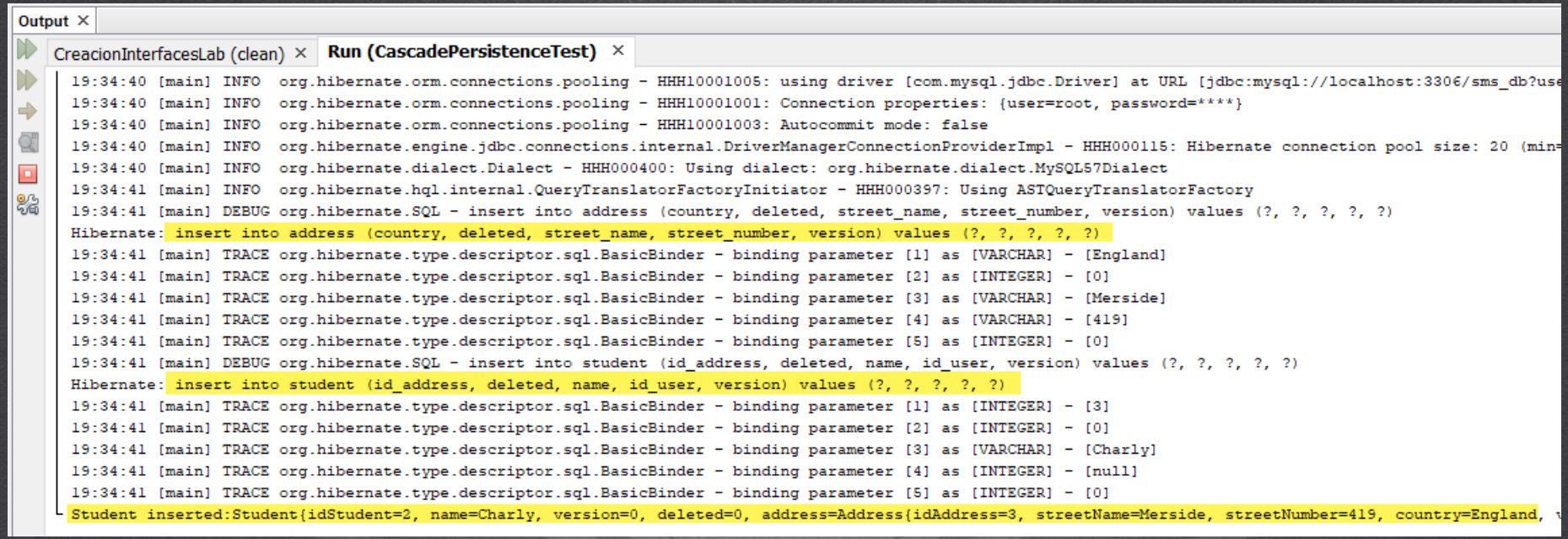
8. EXECUTE CLEAN & BUILD



9. EXECUTE THE CLASS



9. EXECUTE THE CLASS



EXERCISE CONCLUSION

- With this exercise we have seen how to persist cascading an object and its relations with Hibernate / JPA.
- By configuring the Entity object, and its relationships, it is possible to automatically save the Entity object together with its relations, as we have specified in the configuration of the object.
- We also use the log4j2 API to observe the SQLs that are executed when the related entity objects persist in cascade.

ONLINE COURSE

HIBERNATE & JPA

By: Eng. Ubaldo Acosta