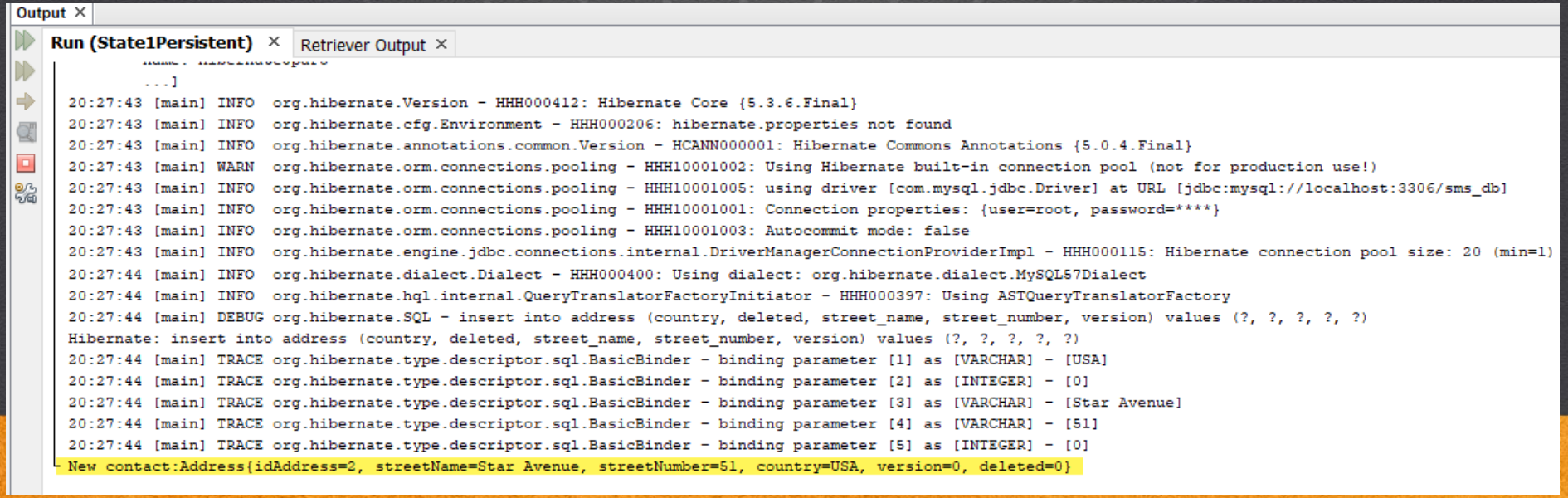
# EXERCISE

# LIFECYCLE IN HIBERNATE/JPA

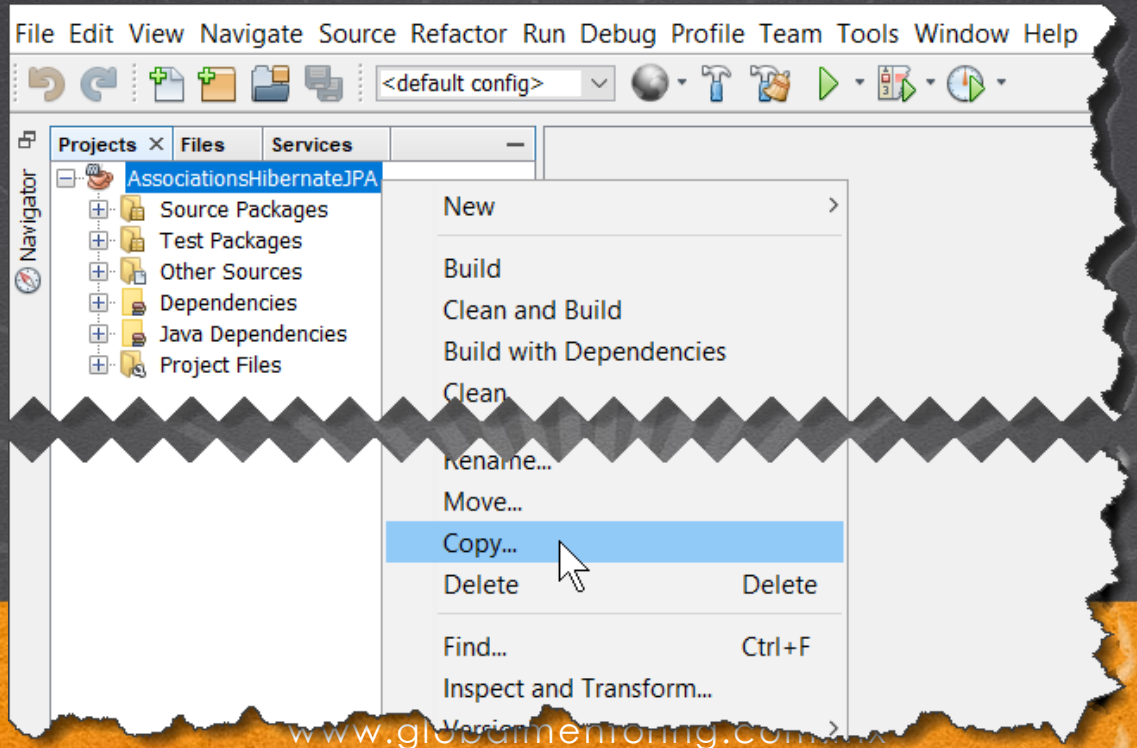
# EXERCISE OBJECTIVE

Implement the Life Cycle of entity objects in Hibernate / JPA.
At the end we will see:



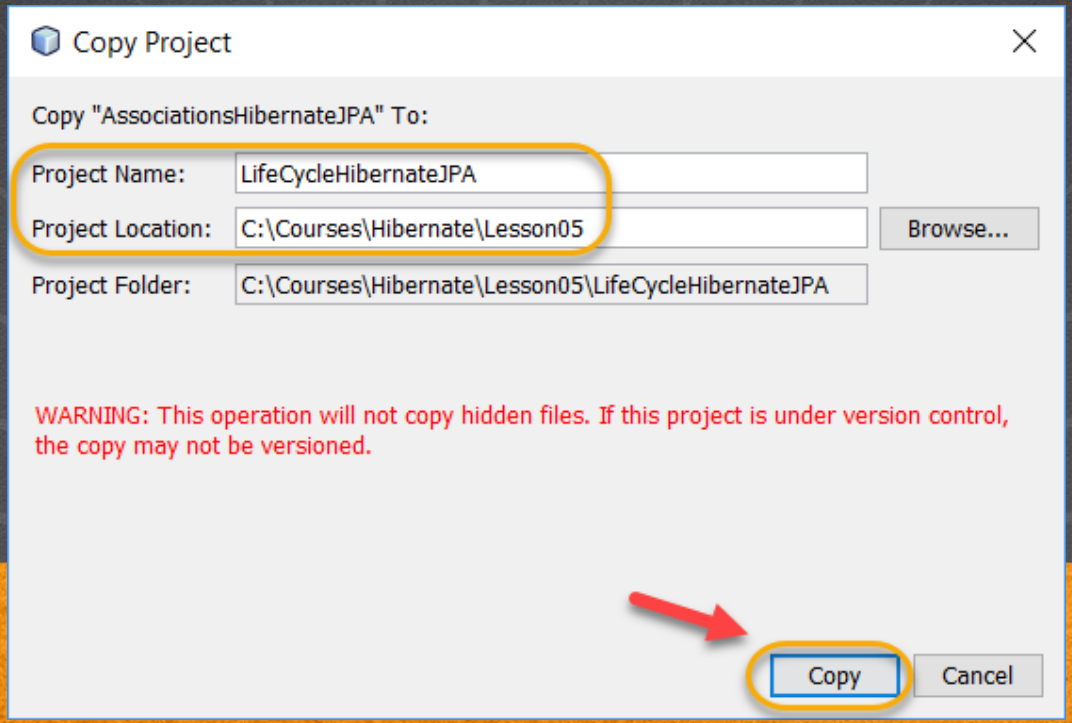
# 1. CREATE A PROJECT

We copy and paste the previous project AsociacionesHibernateJPA:



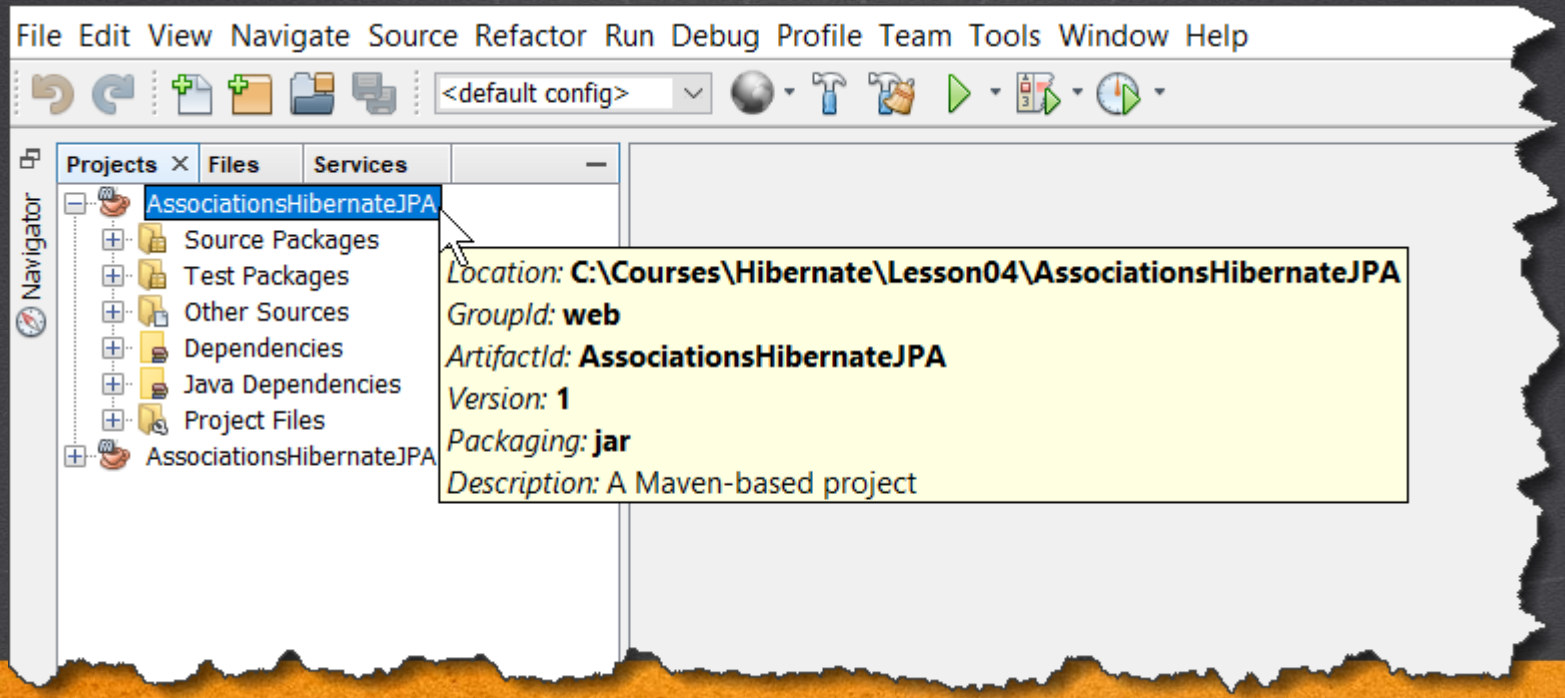
# 1. CREATE A PROJECT

We copy and paste the previous project AsociacionesHibernateJPA:



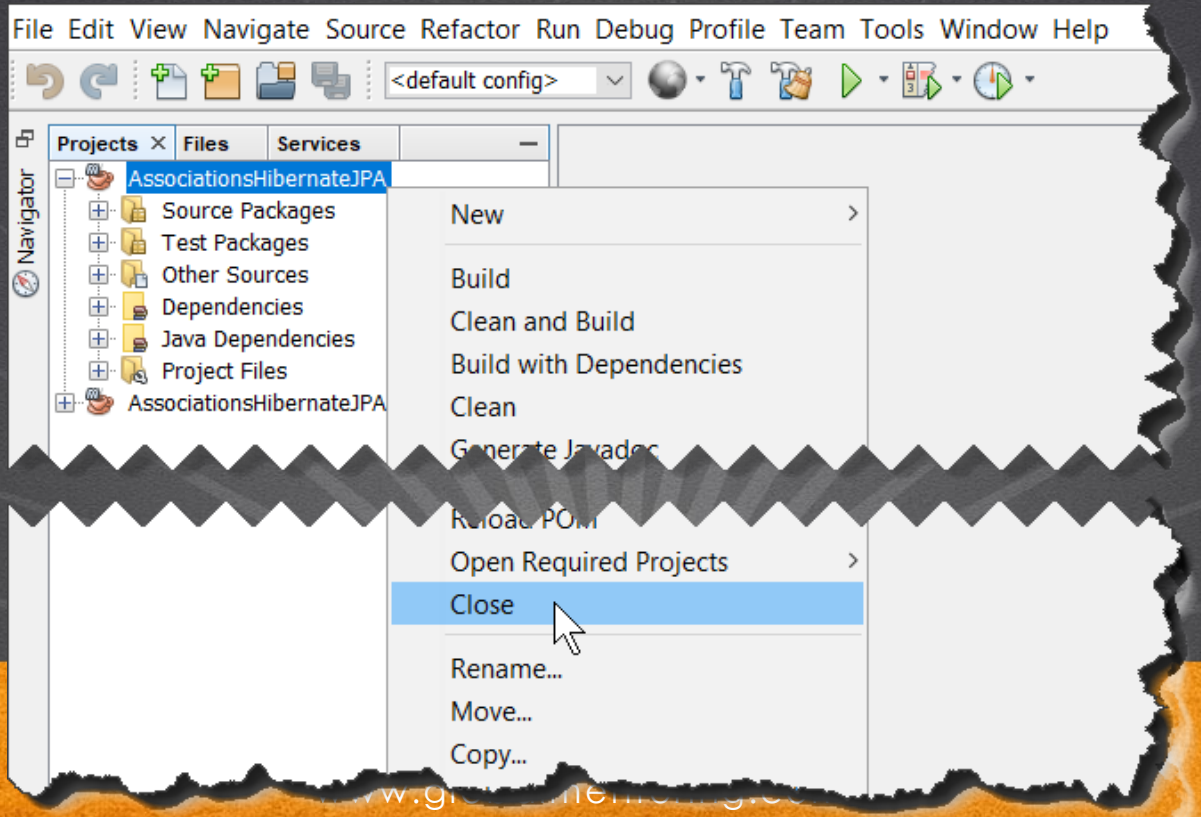
# 2. CLOSE THE PREVIOUS PROJECT

We review the project we want to close:



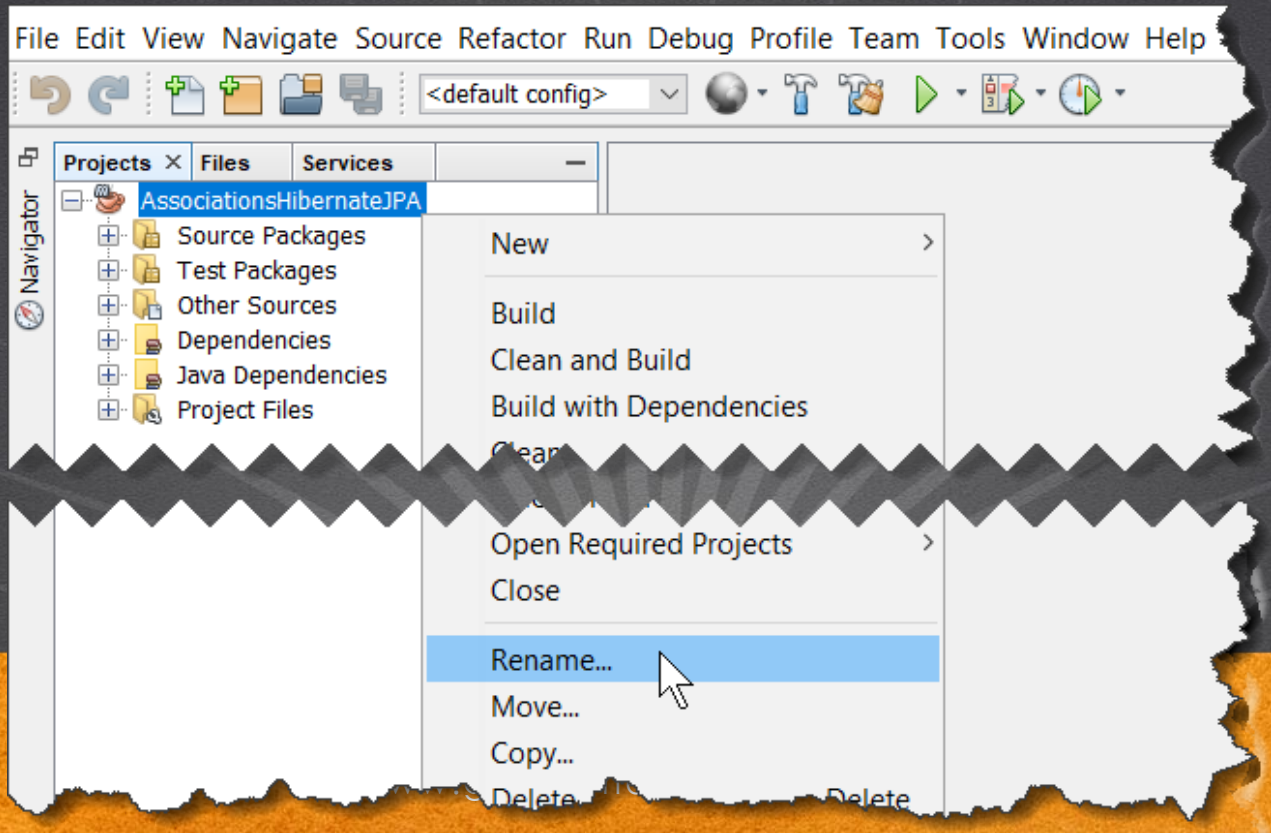
# 2. CLOSE THE PREVIOUS PROJECT

We close the project that we no longer use:



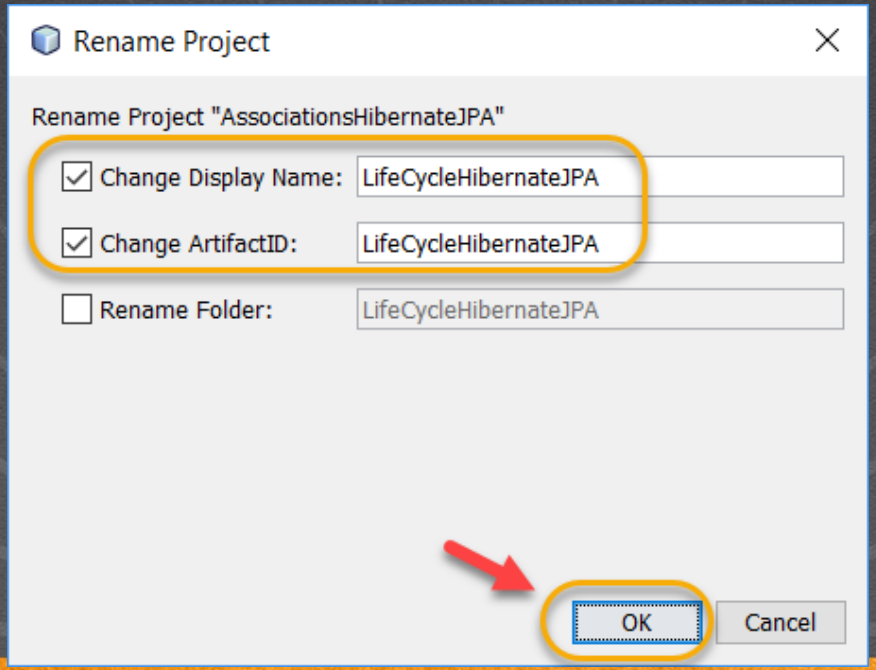
# 3. RENAME THE PROJECT

We rename the project:



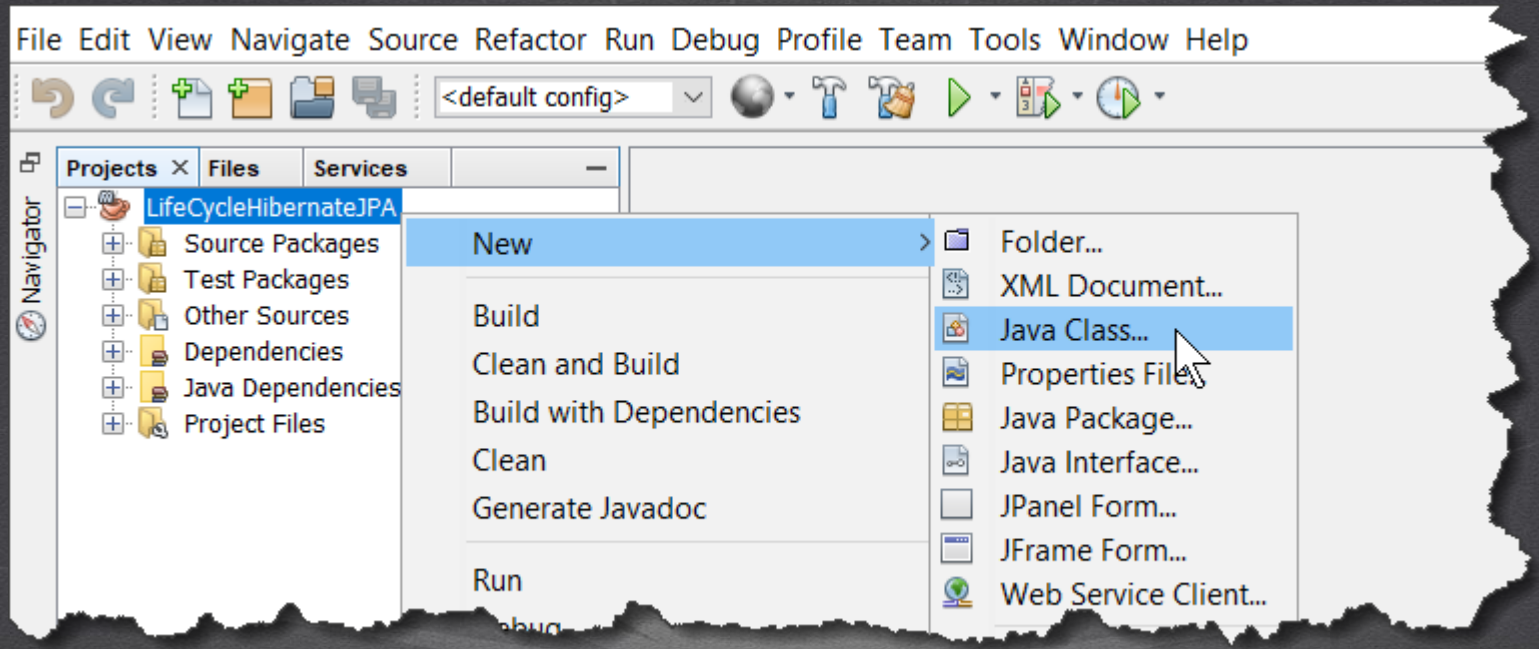
# 3. RENAME THE PROJECT

We rename the project:



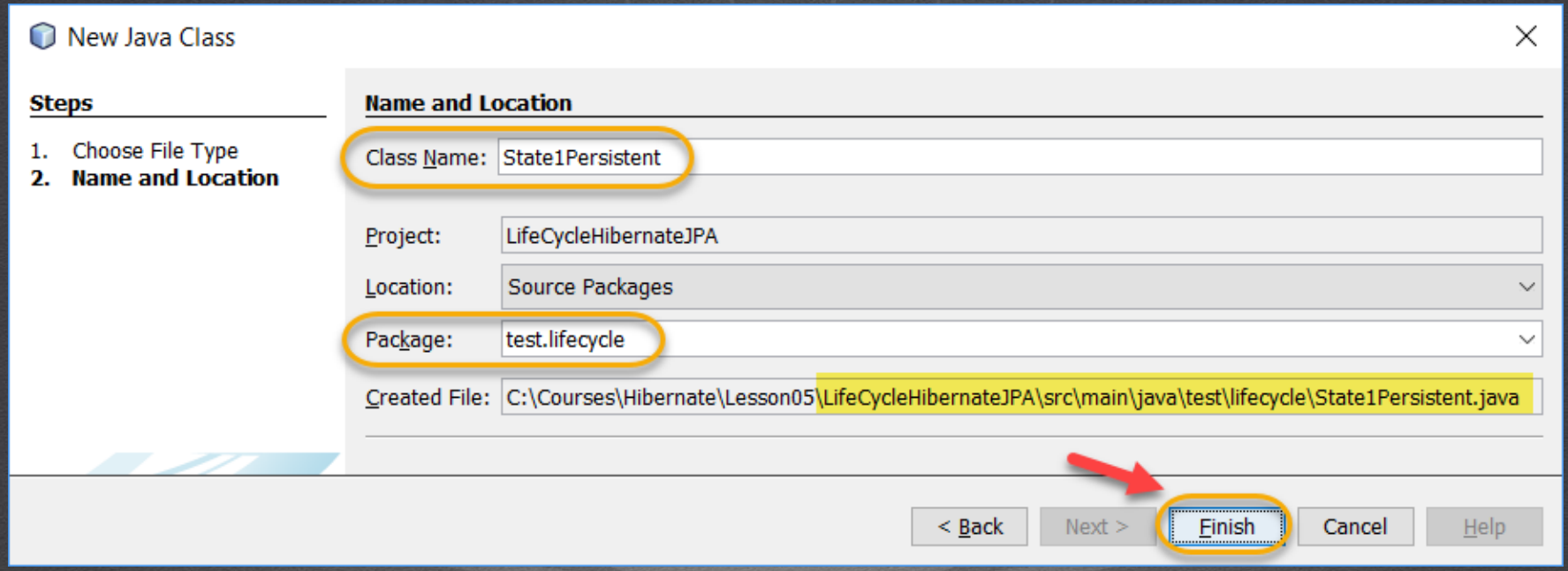
# 4. CREATE A JAVA CLASS

We create a Java class:



# 4. CREATE A JAVA CLASS

We create a Java class:



# 5. MODIFY THE CODE

## State1Persistent.java:

Click to download

```
package test.lifecycle;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import model.Address;

public class State1Persistent {

    public static void main(String[] args) {
        /*We use the JPA Persistence Unit*/
        EntityManagerFactory factory = Persistence.createEntityManagerFactory("HibernateJpaPU");
        EntityManager em = factory.createEntityManager();
        /**
         * Objective: Transient to persistent state
         */

        //1. We create the object (new or transitive state)
        Address address = new Address();
        address.setStreetName("Star Avenue");
        address.setStreetNumber("51");
        address.setCountry("USA");
```

# 5. MODIFY THE CODE

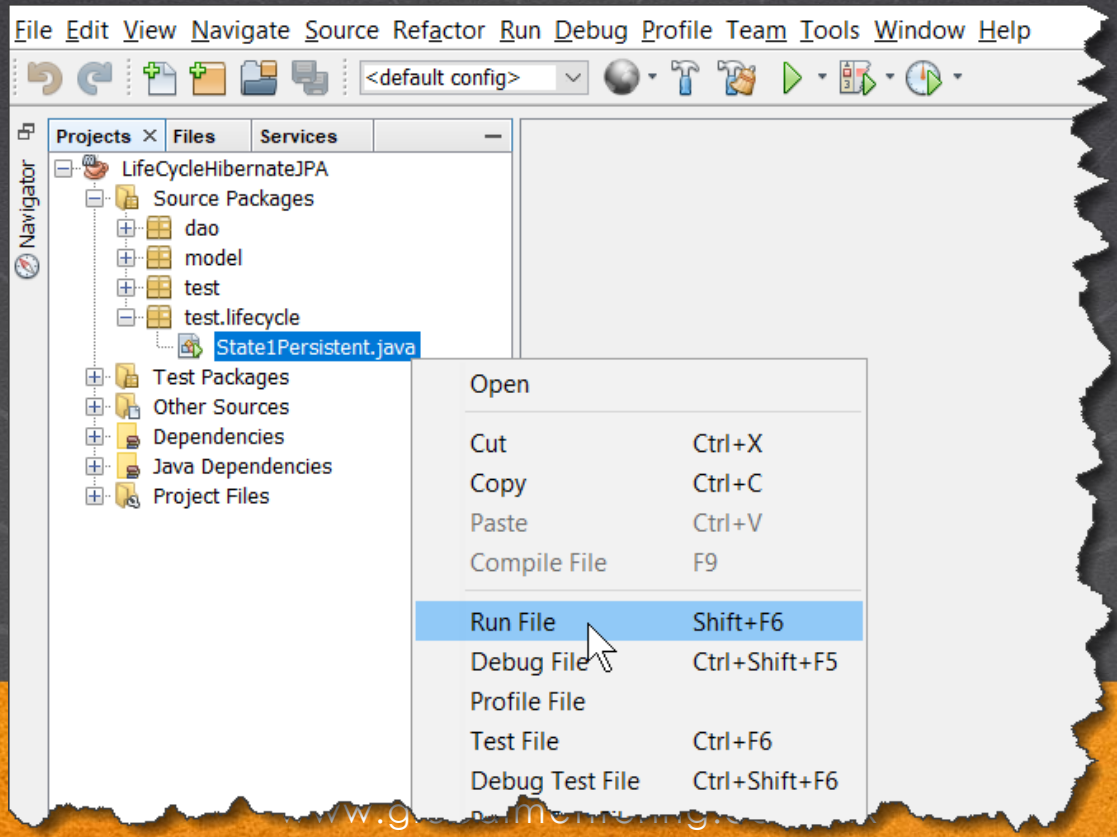
State1Persistent.java:

Click to download

```
        //2. We persist the object (it changes state to persistent)
        //We start the transaction
        //We will not use the DAO classes in these exercises
        try {
            // We start a transaction
            em.getTransaction().begin();
            // Insert the new address
            em.persist(address);
            // We finish the transaction
            em.getTransaction().commit();
        } catch (Exception ex) {
            System.out.println("Error inserting object:" + ex.getMessage());
        } finally {
            if (em != null) {
                em.close();
            }
        }

        //The address object changes from persistent state to detached
        //at the close of the session
        System.out.println("New contact:" + address);
    }
}
```

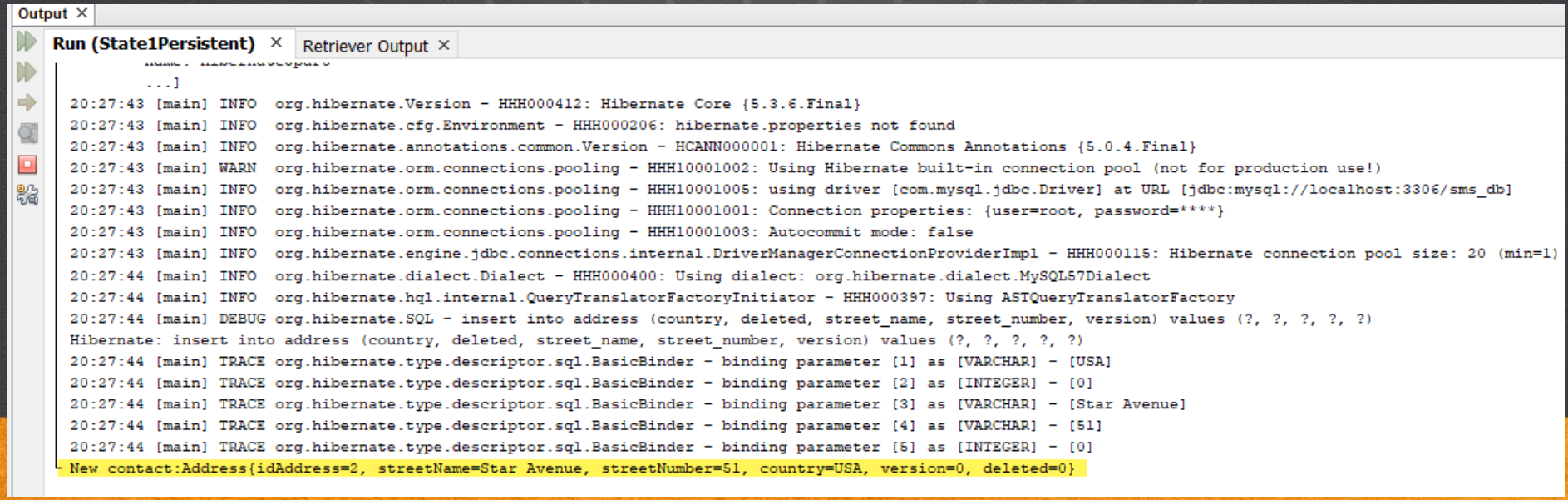
# 6. EXECUTE THE CLASS

Execute the class:



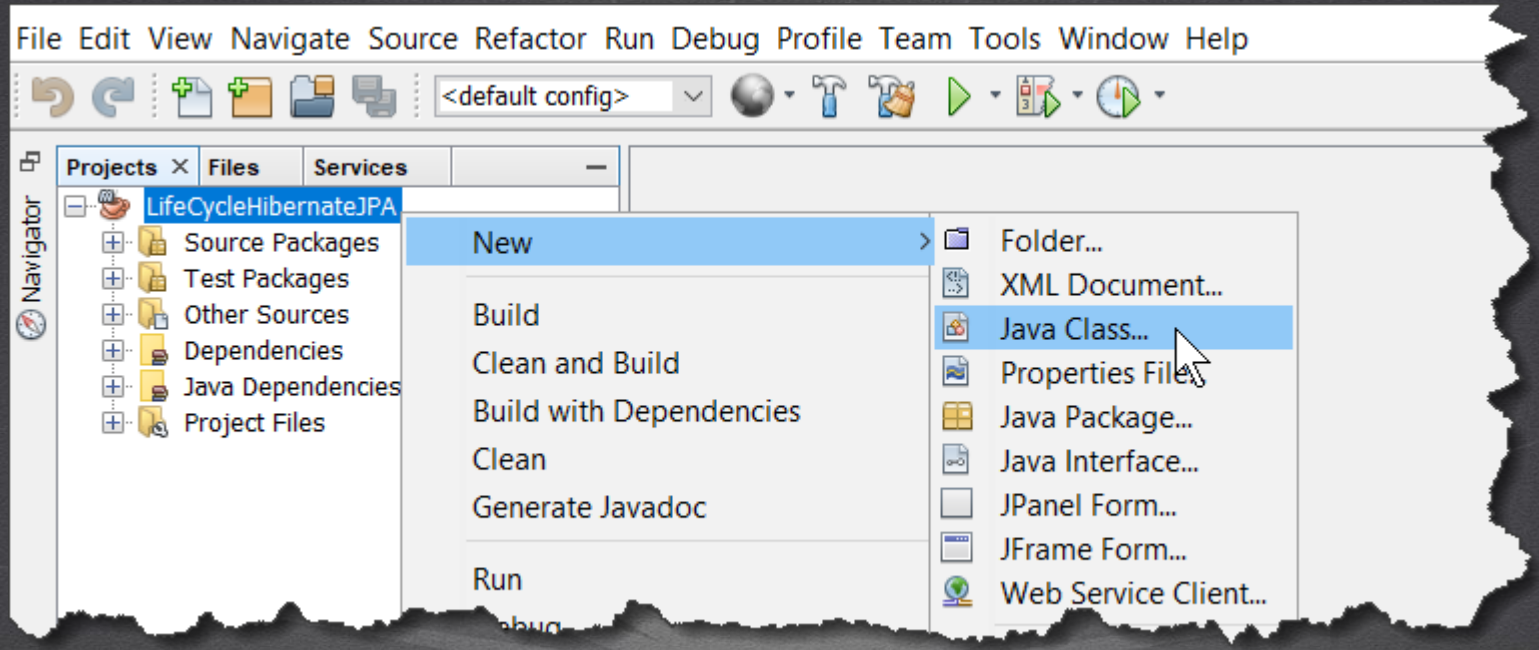
# 6. EXECUTE THE CODE

We observe the result of inserting a new Address type object. The value of idAddress may vary depending on the records that are in the database in the address table:



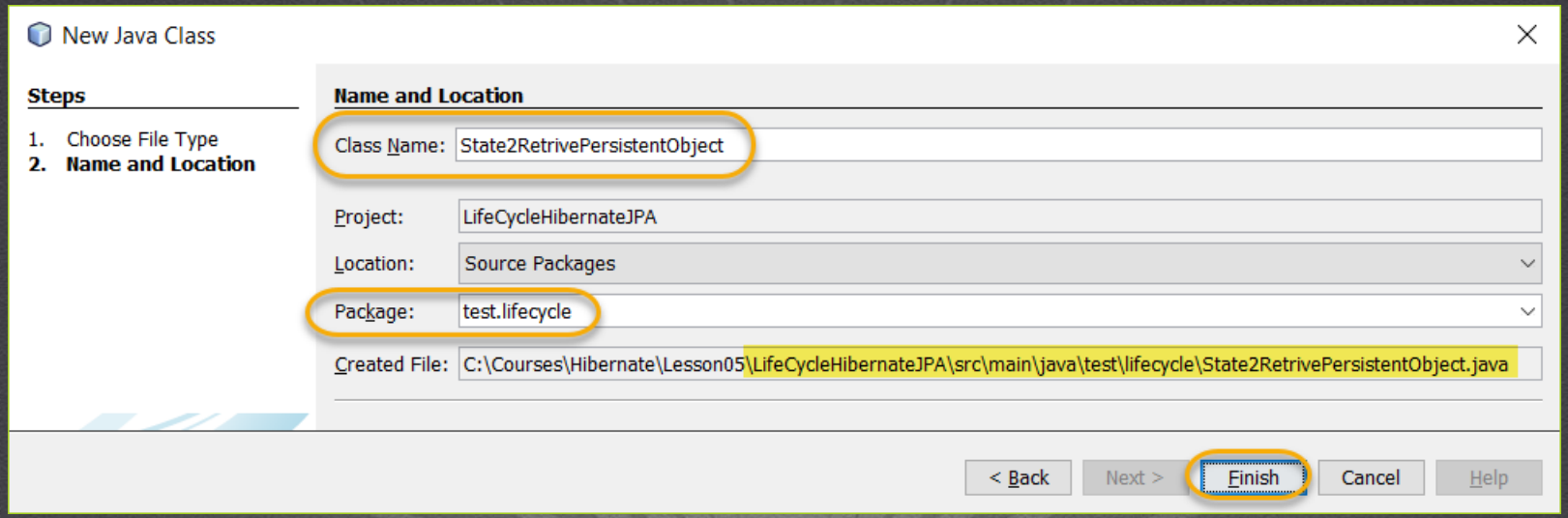
# 7. CREATE A NEW CLASS

Create a new Java class:



# 7. CREATE A JAVA CLASS

Create a Java class:



# 8. MODIFY THE FILE

State2RetrivePersistentObject.java:

Click to download

```
package test.lifecycle;

import javax.persistence.*;
import javax.persistence.Persistence;
import model.Address;

public class State2RetrivePersistentObject {

    public static void main(String[] args) {
        EntityManagerFactory factory = Persistence.createEntityManagerFactory("HibernateJpaPU");
        EntityManager em = factory.createEntityManager();
        /**
         * Objective: Recover a persistent object with Hibernate / JPA
         */

        //There are 2 ways to perform this action
        //1. Method get, return null if you can not find the object
        Address address1 = null;

        try {
            em.getTransaction().begin();
            //We provide the ID to recover, in JPA we use find instead of get
            address1 = (Address) em.find(Address.class, 2);
            System.out.println("Address retrieved with find:" + address1);
            em.getTransaction().commit(); //we make flush
        } catch (Exception e) {
            em.getTransaction().rollback();
            e.printStackTrace(System.out);
        }
```

# 8. MODIFY THE FILE

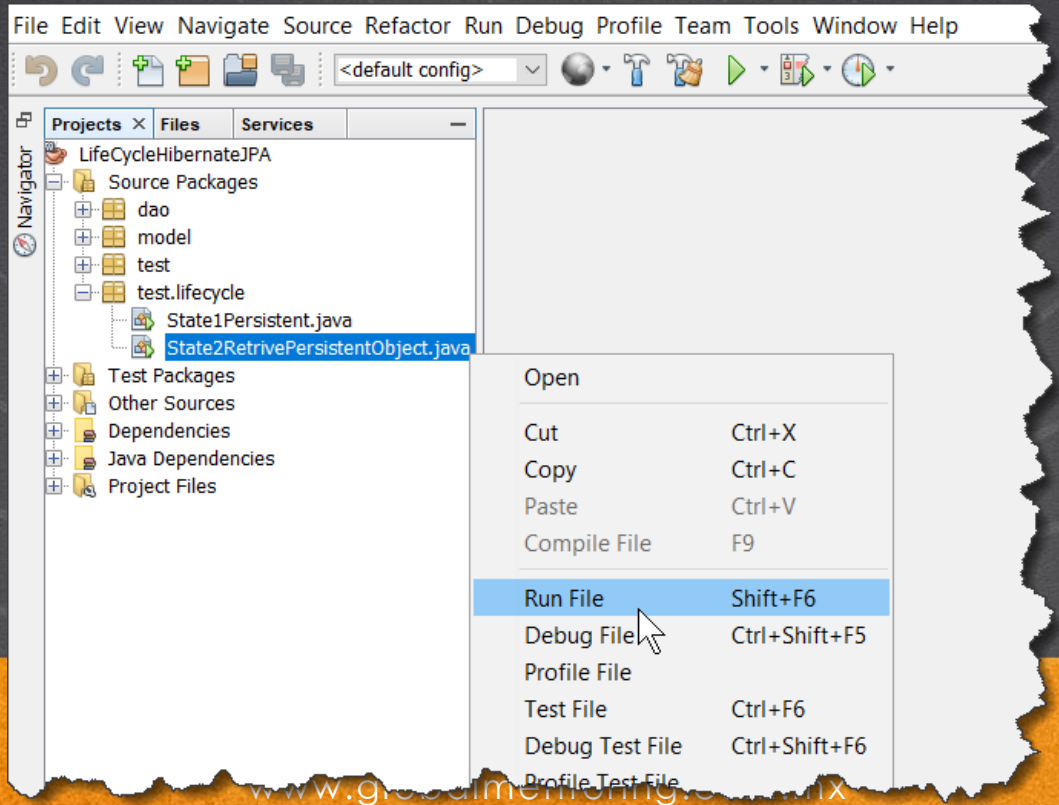
State2RetrivePersistentObject.java:

Click to download

```
        //The object of address1 changes to edo. detached at the close of the session

        //2. Method load, return ObjectNotFoundException
        //if you can not find the id provided
        Address address2 = null;
        //We create a second transaction
        try {
            em.getTransaction().begin();

            //We provide the id to recover, it does not get relationships (lazy)
            //In JPA we have getReference as another option
            address2 = (Address) em.getReference(Address.class, 2);
            System.out.println("Address retrieved with getReference:" + address2);
            em.getTransaction().commit();
        } catch (Exception e) {
            em.getTransaction().rollback();
            e.printStackTrace(System.out);
        } finally {
            if (em != null) {
                em.close();
            }
        }

        //The address2 object changes to detached state
        //when closing transaction 2
    }
}
```

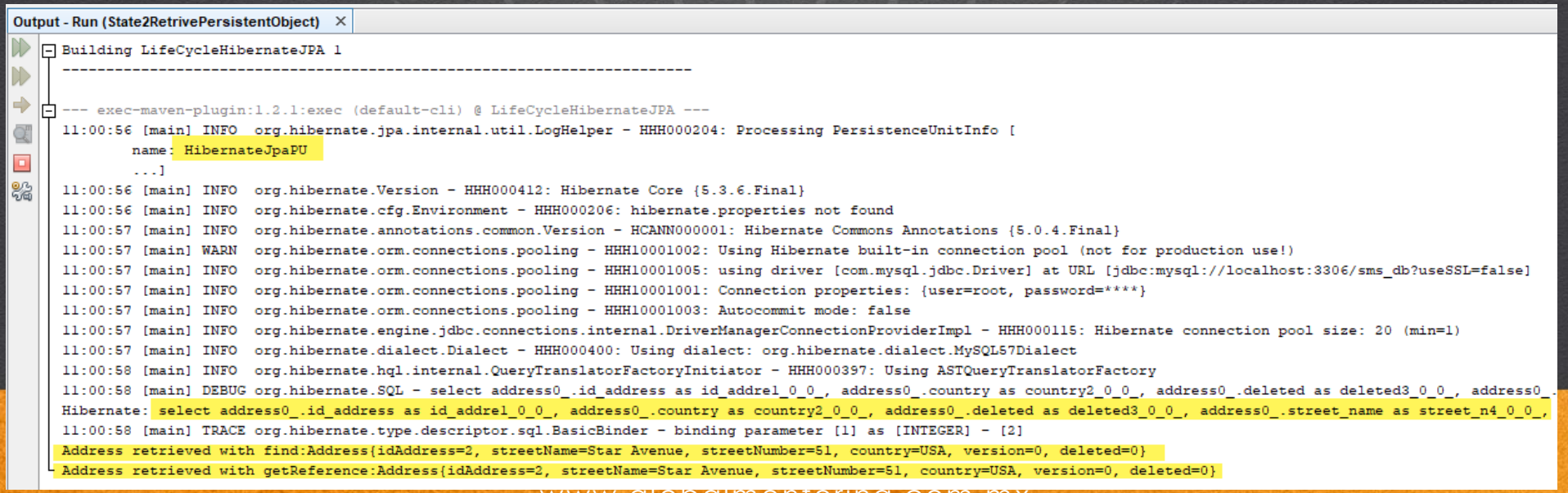
# 9. EXECUTE THE CODE

Execute the class:



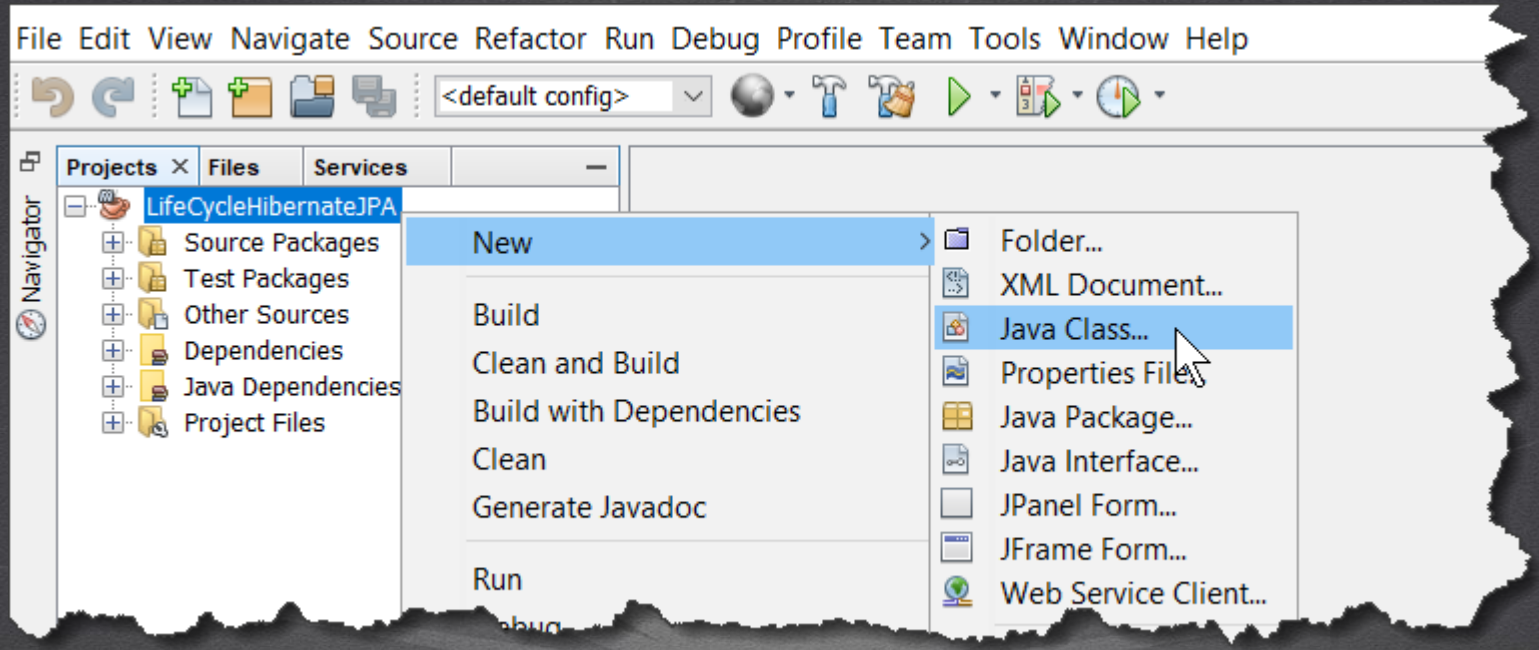
# 9. EXECUTE THE CODE

We observe the result of executing our class. The idAddress to search may not exist, so the value must be adapted to an existing value.



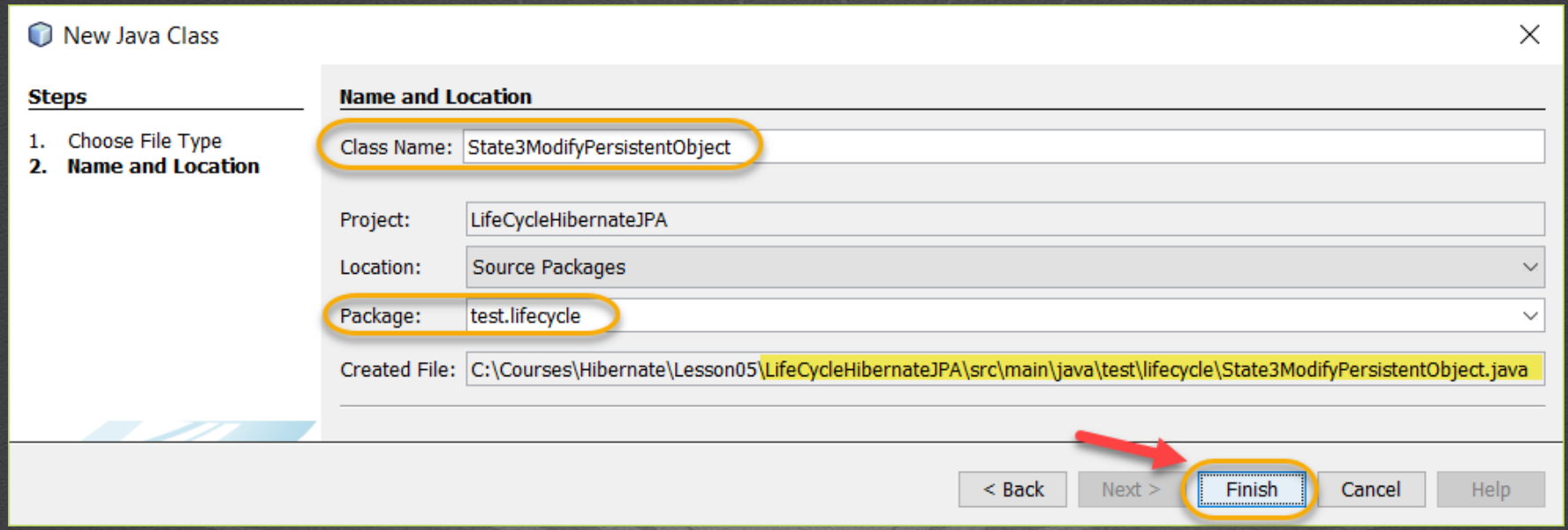
# 10. CREATE A NEW JAVA CLASS

Create a new class:



# 10. CREATE A JAVA CLASS

## Create a Java Class:



# 11. MODIFY THE CODE

State3ModifyPersistentObject.java:

Click to download

```
package test.lifecycle;

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

public class State3ModifyPersistentObject {

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

        /**
         * Objective: Modify a persistent object (reattaching) Move from a
         * detached to persistent state
         */
        //We recover a persistent object
        Address address = null;
        try {
            em.getTransaction().begin();
            //Identifier to recover, you can also use merge to retrieve an object
            address = (Address) em.find(Address.class, 1);
            em.getTransaction().commit(); //hacemos flush
        } catch (Exception e) {
            em.getTransaction().rollback();
            e.printStackTrace(System.out);
        }
```

# 11. MODIFY THE CODE

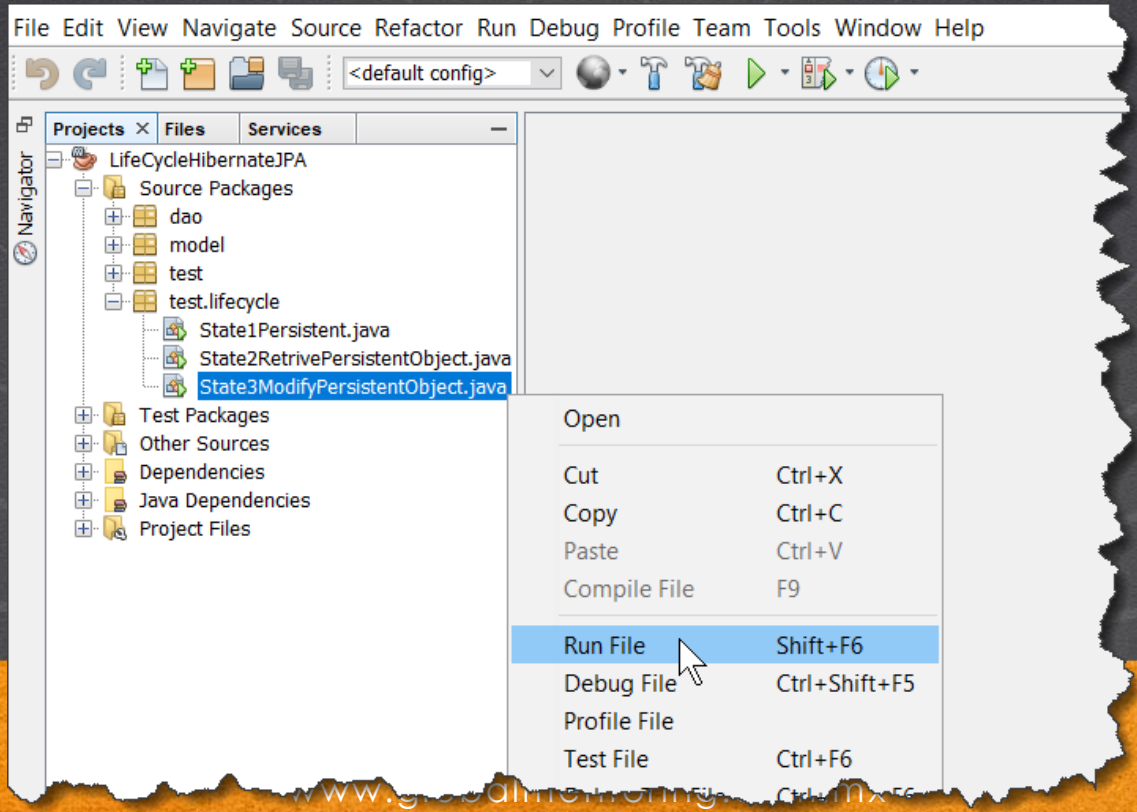
State3ModifyPersistentObject.java:

Click to download

```
        //We print the values of the recovered object
        //state detached
        System.out.println("Address recovered:" + address);

        //We modify the object outside of a transaction
        if (address != null) {
            address.setStreetName("new street");
        }

        //We return to save the object in a new transaction
        //We change the state from detached to persistent (reattaching)
        try {
            em.getTransaction().begin();
            em.merge(address);
            em.getTransaction().commit(); //we make flush
        } catch (Exception e) {
            em.getTransaction().rollback();
            e.printStackTrace(System.out);
        } finally {
            if (em != null) {
                em.close();
            }
        }
        System.out.println("Modified address object:" + address);
        //The address object changes to detached status when closing session
    }
}
```

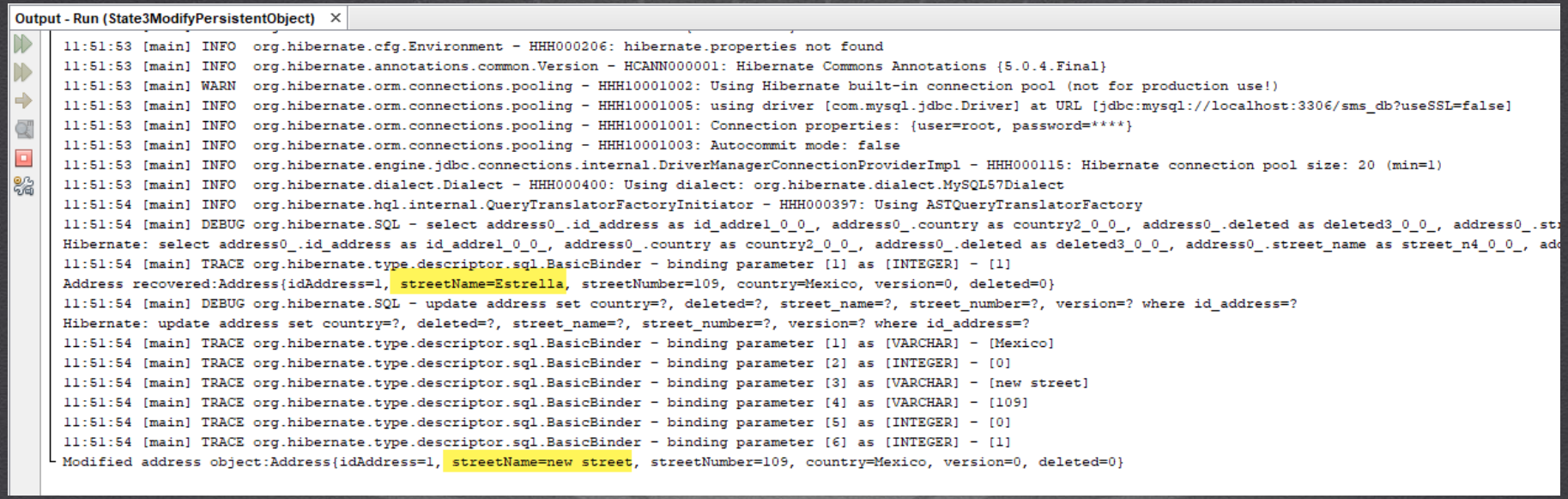
# 12. EXECUTE THE CLASS

Execute the Java class:



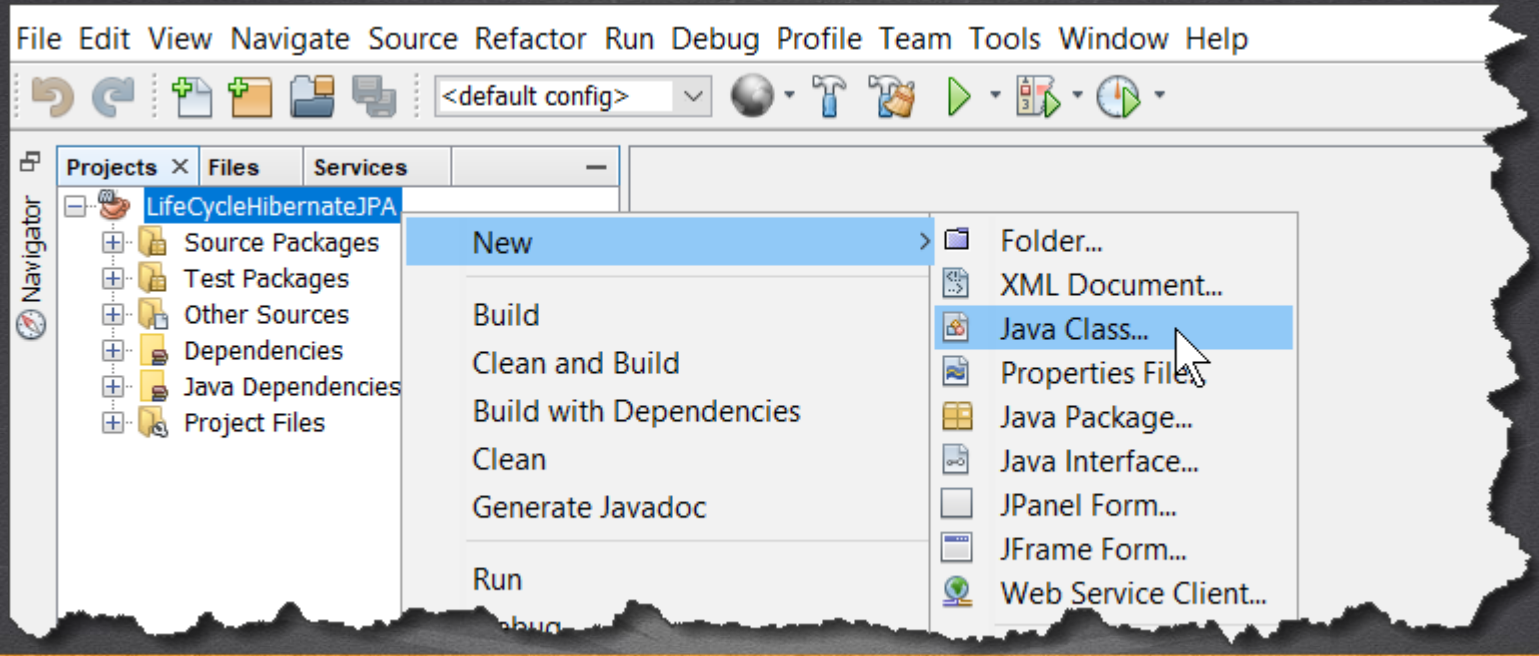
# 12. EXECUTE THE JAVA CLASS

Execute the class:



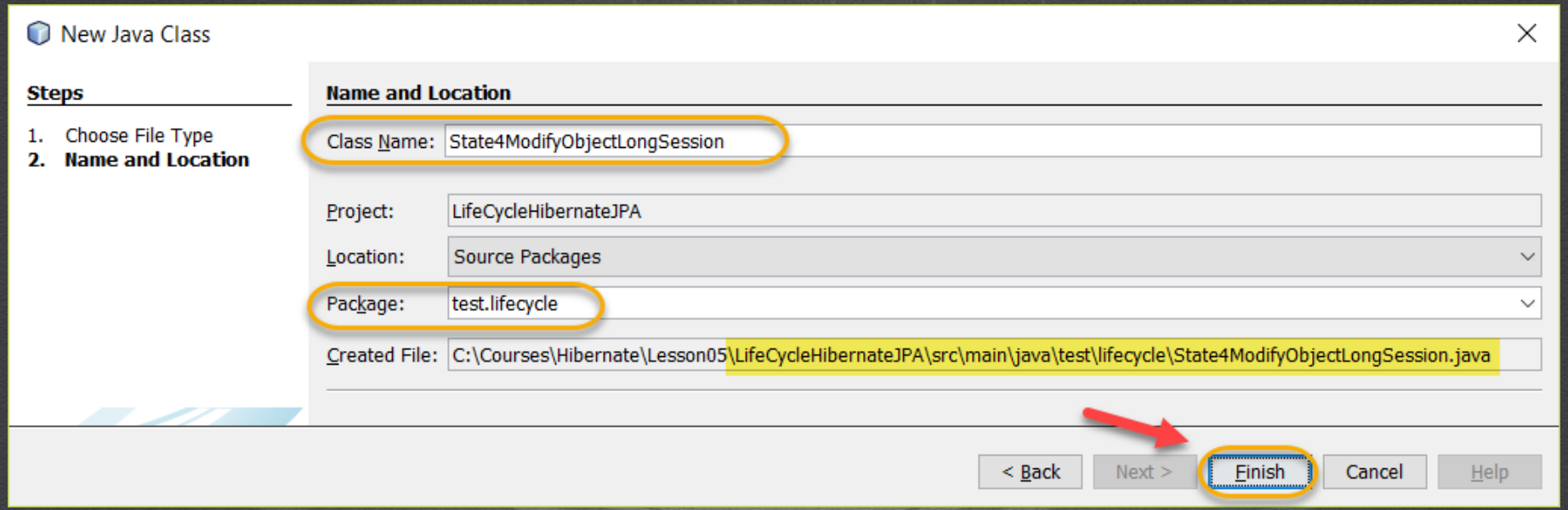
# 13. CREATE A NEW JAVA CLASS

Create a new Java class:



# 13. CREATE A JAVA CLASS

Create a Java class:



# 14. MODIFY THE CODE

State4ModifyObjectLongSession.java:

Click to download

```
package test.lifecycle;

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

public class State4ModifyObjectLongSession {

    public static void main(String[] args) {
        /*We use the JPA Persistence Unit */
        EntityManagerFactory fabrica = Persistence.createEntityManagerFactory("HibernateJpaPU");
        EntityManager em = fabrica.createEntityManager();

        //We recover a persistent object
        Address address = null;
```

# 14. MODIFY THE CODE

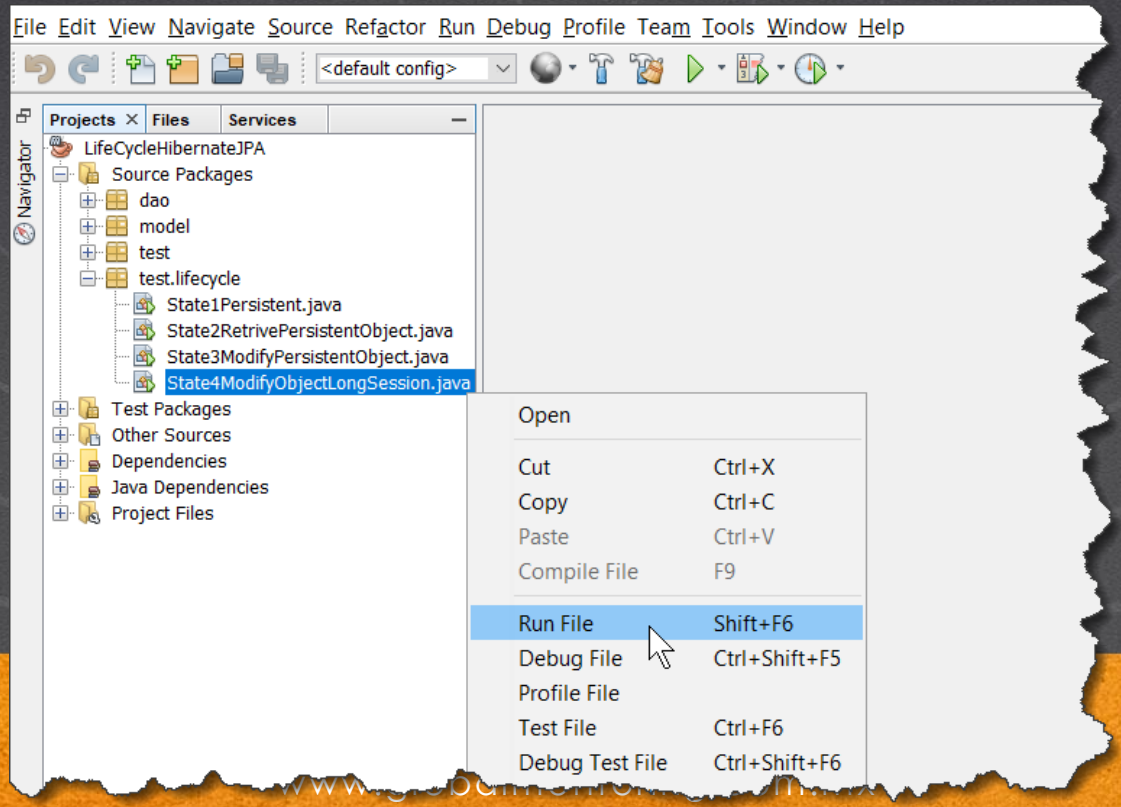
State4ModifyObjectLongSession.java:

Click to download

```
        try {
            em.getTransaction().begin();
            //Identifier to recover, you can also use merge to retrieve an object
            address = (Address) em.find(Address.class, 1);
            System.out.println("Address unmodified:" + address);
            //We modify the object in the same transaction
            address.setStreetNumber("555");
            em.getTransaction().commit(); //hacemos flush
        } catch (Exception e) {
            em.getTransaction().rollback();
            e.printStackTrace(System.out);
        } finally {
            if (em != null) {
                em.close();
            }
        }

        //We print the values (status detached)
        System.out.println("Address recovered:" + address);
    }

}
```

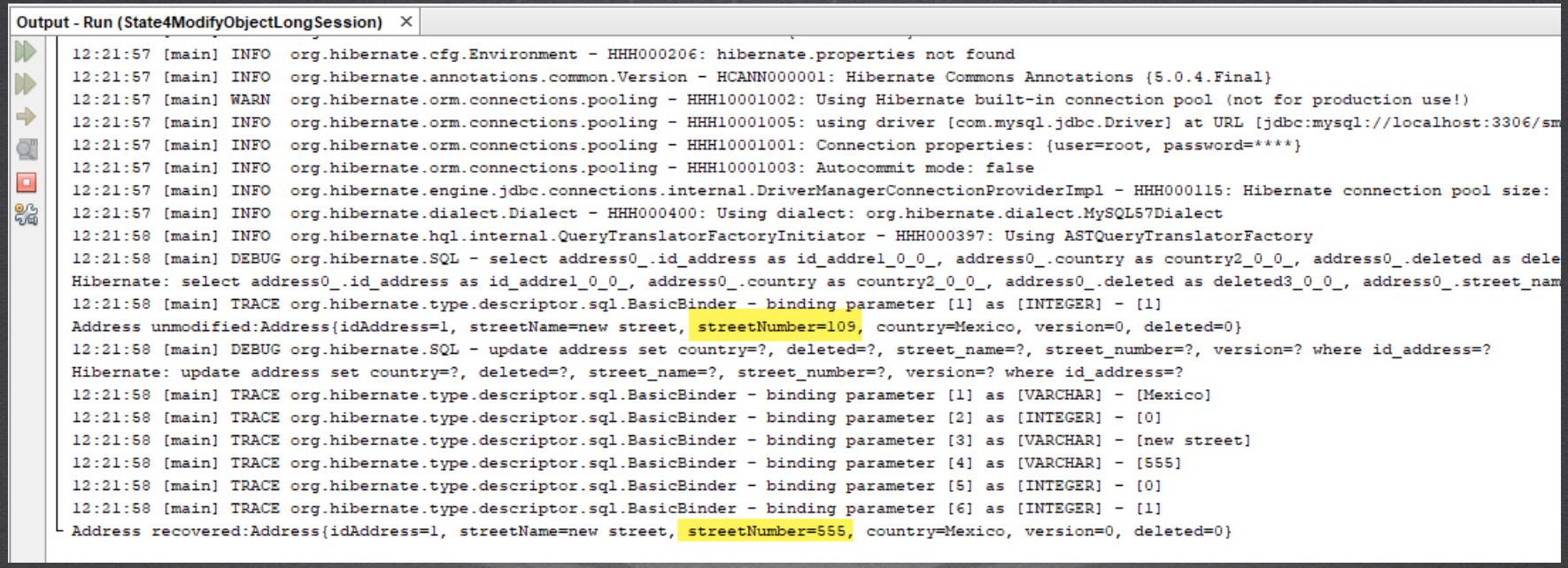
# 15. EXECUTE THE JAVA CLASS

Execute the Java class:



# 15. EXECUTE THE JAVA CLASS

Execute the Java class:



# EXECISE CONCLUSION

- With this exercise we have seen the life cycle of entity objects supported by the JPA API.
- The life cycle of Entity objects in Hibernate / JPA is very important to know, since it depends on that we can correctly perform the operations on our Entity objects and at the end on our records in the respective database table.
- With this we conclude the theme of life cycle in Hibernate / JPA.

ONLINE COURSE

# HIBERNATE & JPA

By: Eng. Ubaldo Acosta