Hibernate Annotations

A simple way to map entities without XML

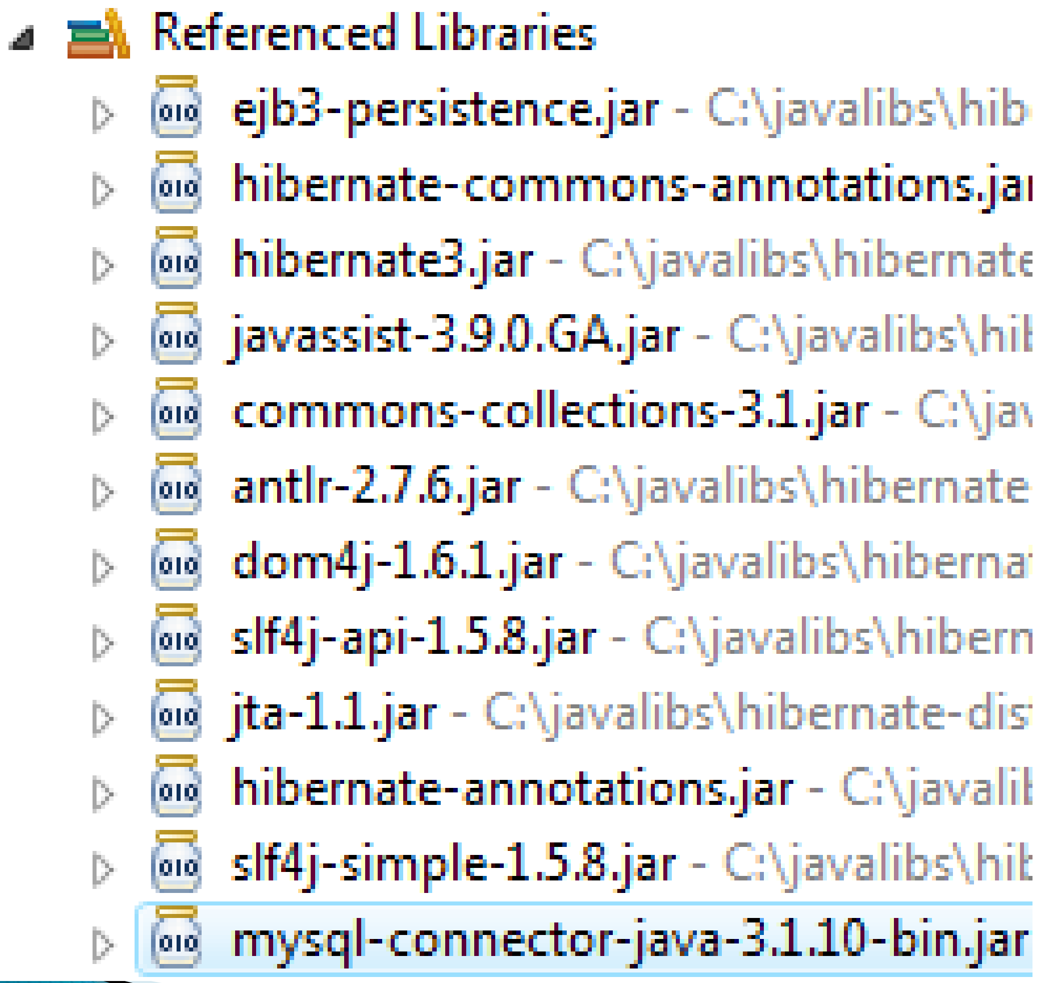
EHSDI

eBuzima

Hibernate Annotations

- Annotations allow you to map entities and their relationships without xml files.
- All the mapping information will be in the code for each domain object

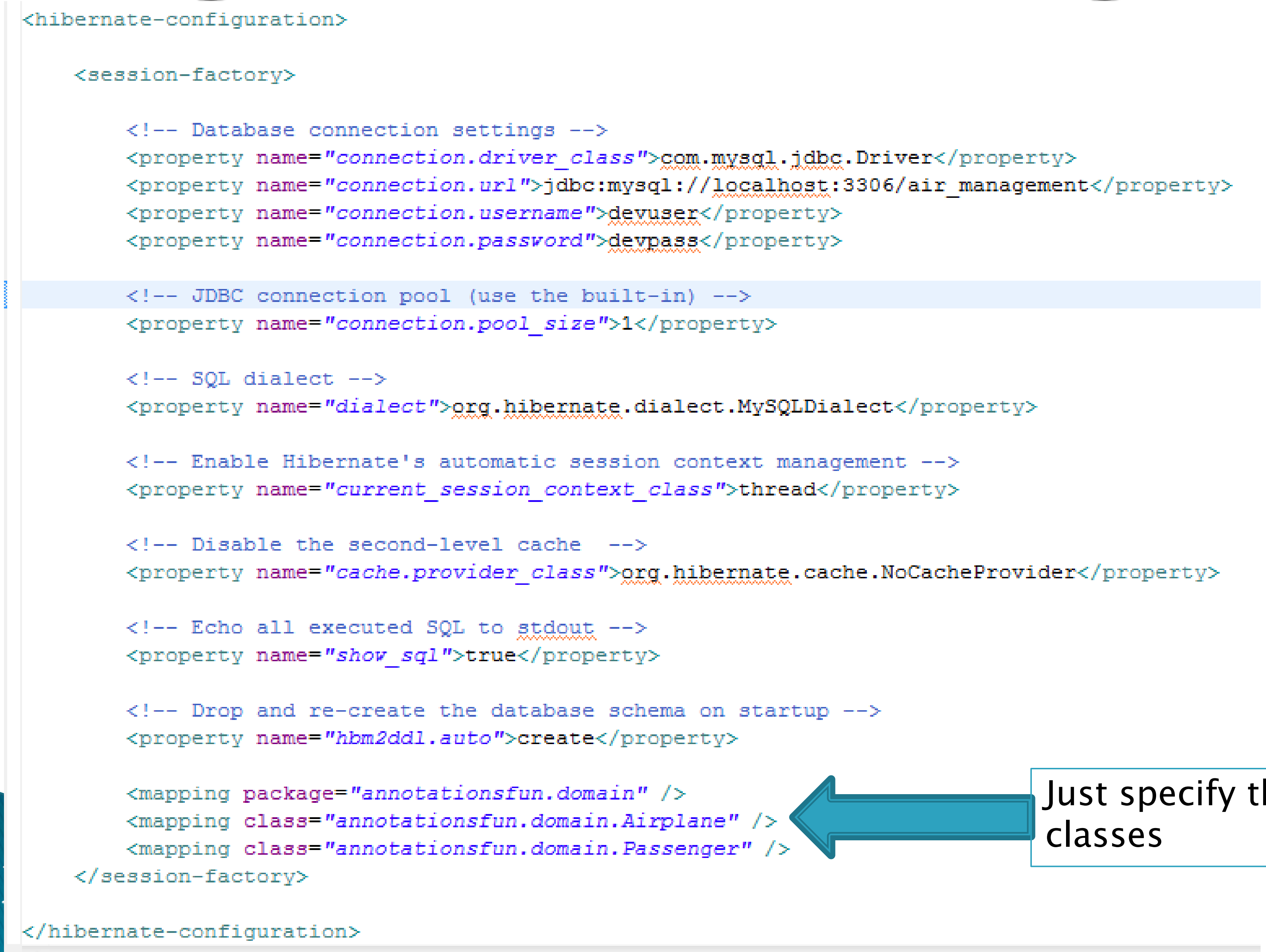
Required Libraries



Referenced Libraries
- ejb3-persistence.jar - C:\javalibs\hib
- hibernate-commons-annotations.jar
- hibernate3.jar - C:\javalibs\hibernate
- javassist-3.9.0.GA.jar - C:\javalibs\hil
- commons-collections-3.1.jar - C:\ja
- antlr-2.7.6.jar - C:\javalibs\hibernate
- dom4j-1.6.1.jar - C:\javalibs\hiberna
- slf4j-api-1.5.8.jar - C:\javalibs\hibern
- jta-1.1.jar - C:\javalibs\hibernate-dis
- hibernate-annotations.jar - C:\javalil
- slf4j-simple-1.5.8.jar - C:\javalibs\hil
- mysql-connector-java-3.1.10-bin.jar

EHSDI
eBuzima

Configuration File (hibernate.cfg.xml)

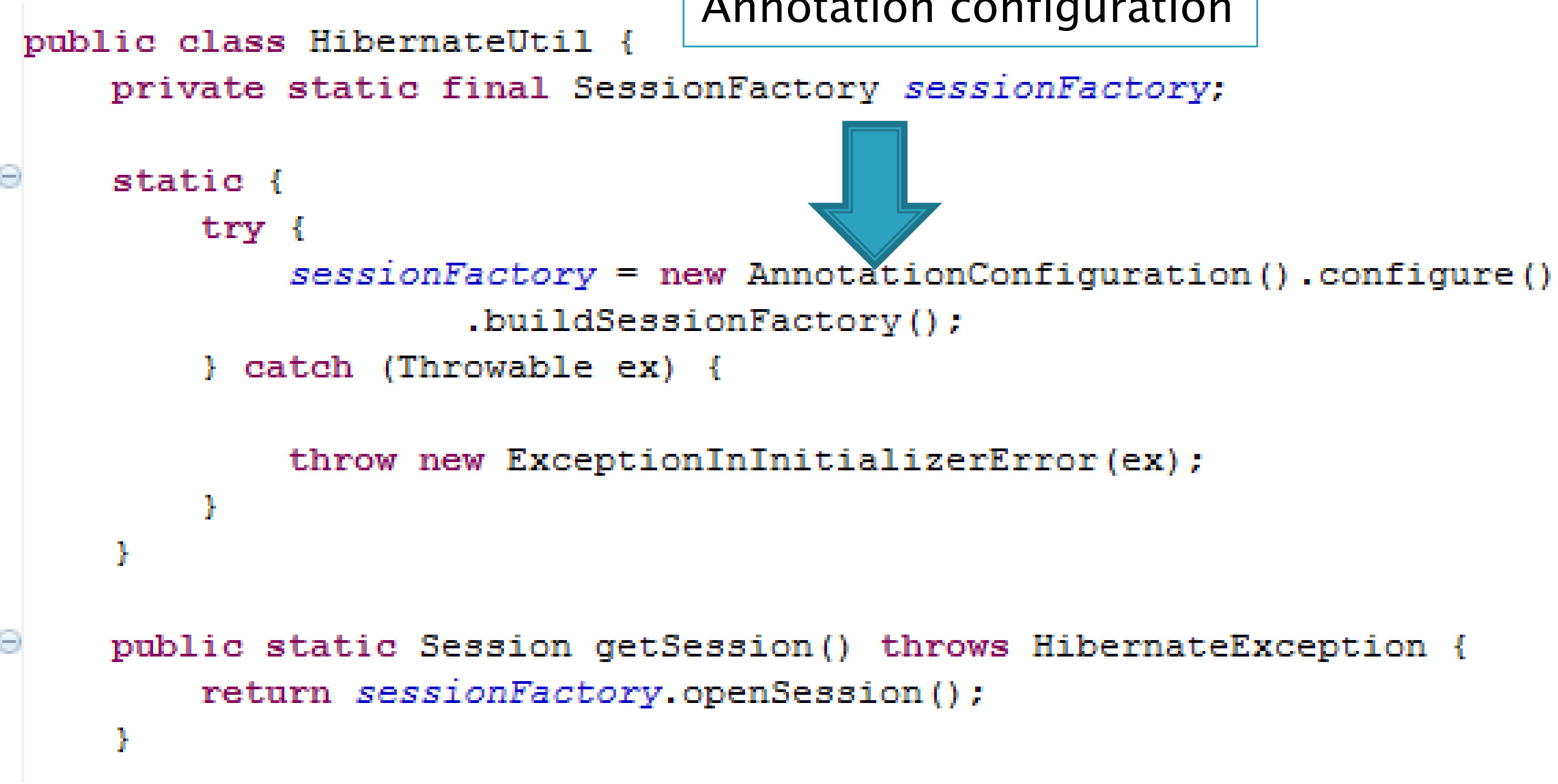


```xml
<hibernate-configuration>

    <session-factory>

        <!-- Database connection settings -->
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/air_management</property>
        <property name="connection.username">devuser</property>
        <property name="connection.password">devpass</property>

        <!-- JDBC connection pool (use the built-in) -->
        <property name="connection.pool_size">1</property>

        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>

        <!-- Enable Hibernate's automatic session context management -->
        <property name="current_session_context_class">thread</property>

        <!-- Disable the second-level cache  -->
        <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="show_sql">true</property>

        <!-- Drop and re-create the database schema on startup -->
        <property name="hbm2ddl.auto">create</property>

        <mapping package="annotationsfun.domain" />
        <mapping class="annotationsfun.domain.Airplane" />
        <mapping class="annotationsfun.domain.Passenger" />
    </session-factory>

</hibernate-configuration>
```

Just specify the classes

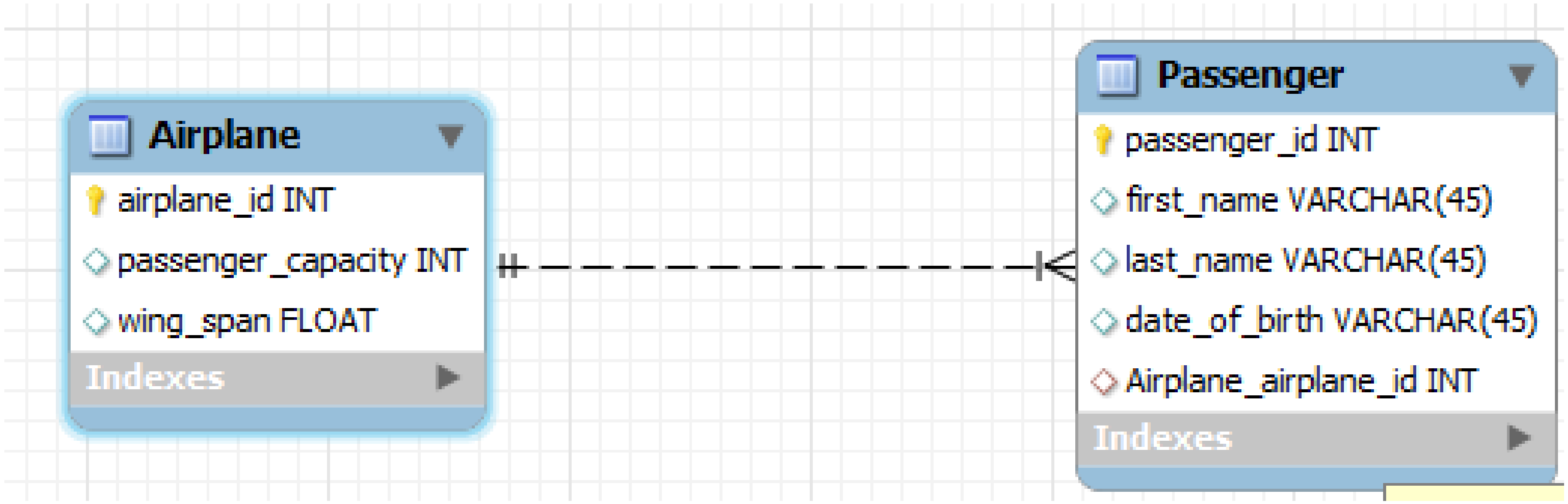
HibernateUtil

Annotation configuration

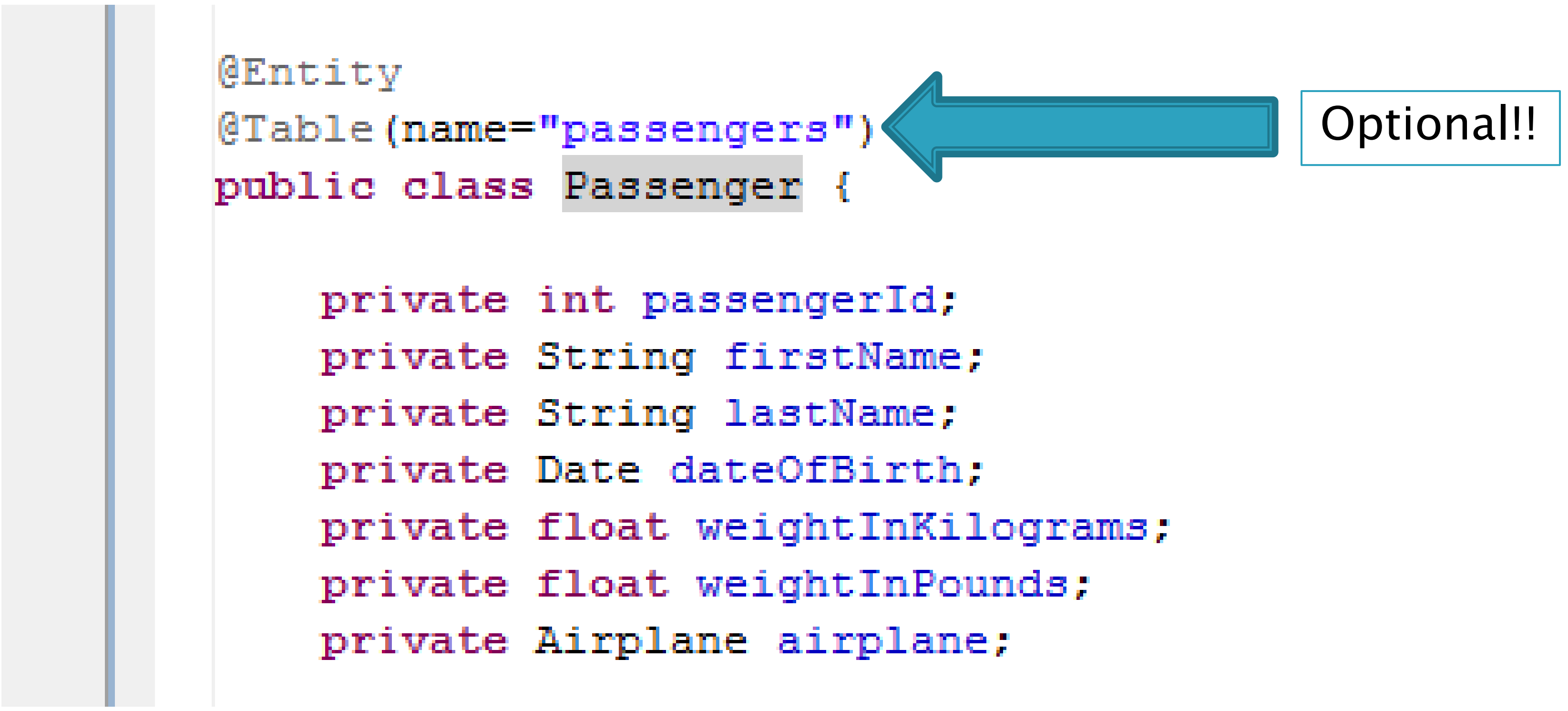


```java
public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {
            sessionFactory = new AnnotationConfiguration().configure()
                    .buildSessionFactory();
        } catch (Throwable ex) {

            throw new ExceptionInInitializerError(ex);
        }
    }

    public static Session getSession() throws HibernateException {
        return sessionFactory.openSession();
    }
```

Entity Mapping!

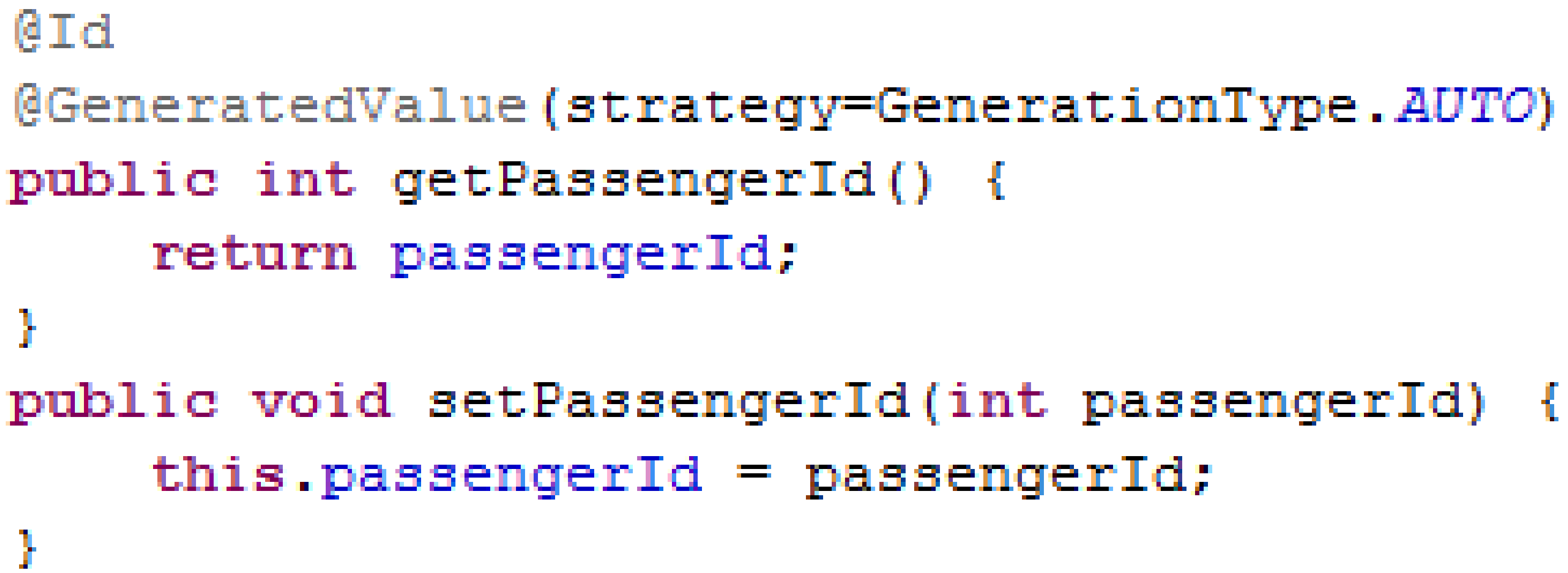


Mapping Entities to Tables



```
@Entity
@Table(name="passengers")
public class Passenger {

    private int passengerId;
    private String firstName;
    private String lastName;
    private Date dateOfBirth;
    private float weightInKilograms;
    private float weightInPounds;
    private Airplane airplane;
```

Mapping IDs



```
@Id
@GeneratedValue(strategy=GenerationType.AUTO)
public int getPassengerId() {
	return passengerId;
}
public void setPassengerId(int passengerId) {
	this.passengerId = passengerId;
}
```

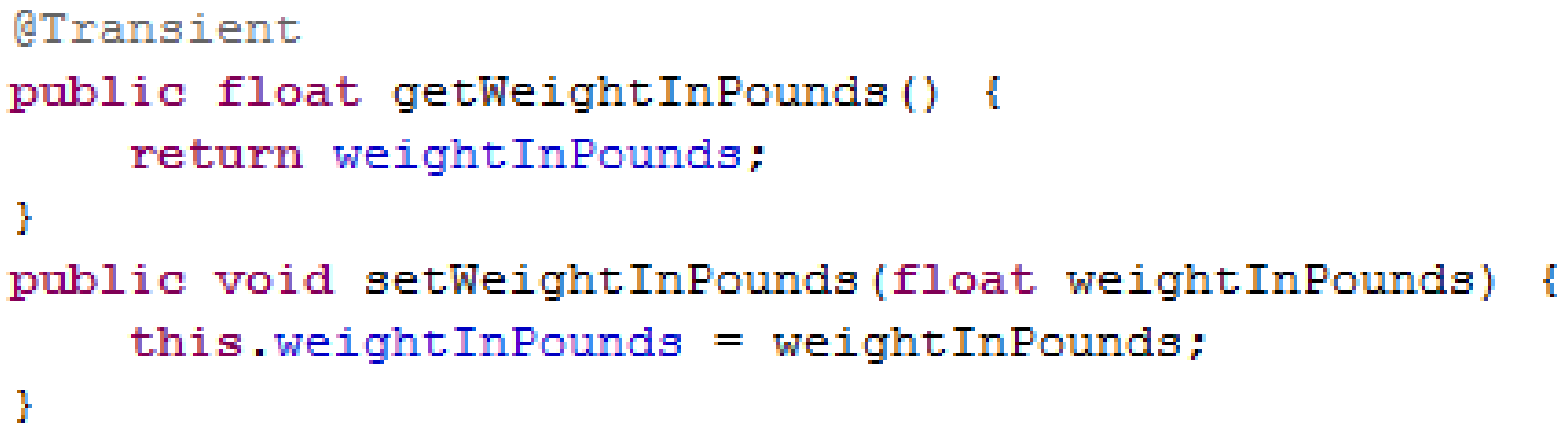
Mapping Columns

```
@Column(name="flight_name")
public float getWeightInKilograms() {
    return weightInKilograms;
}
public void setWeightInKilograms(float weightInKilograms) {
    this.weightInKilograms = weightInKilograms;
}
```

Optional!

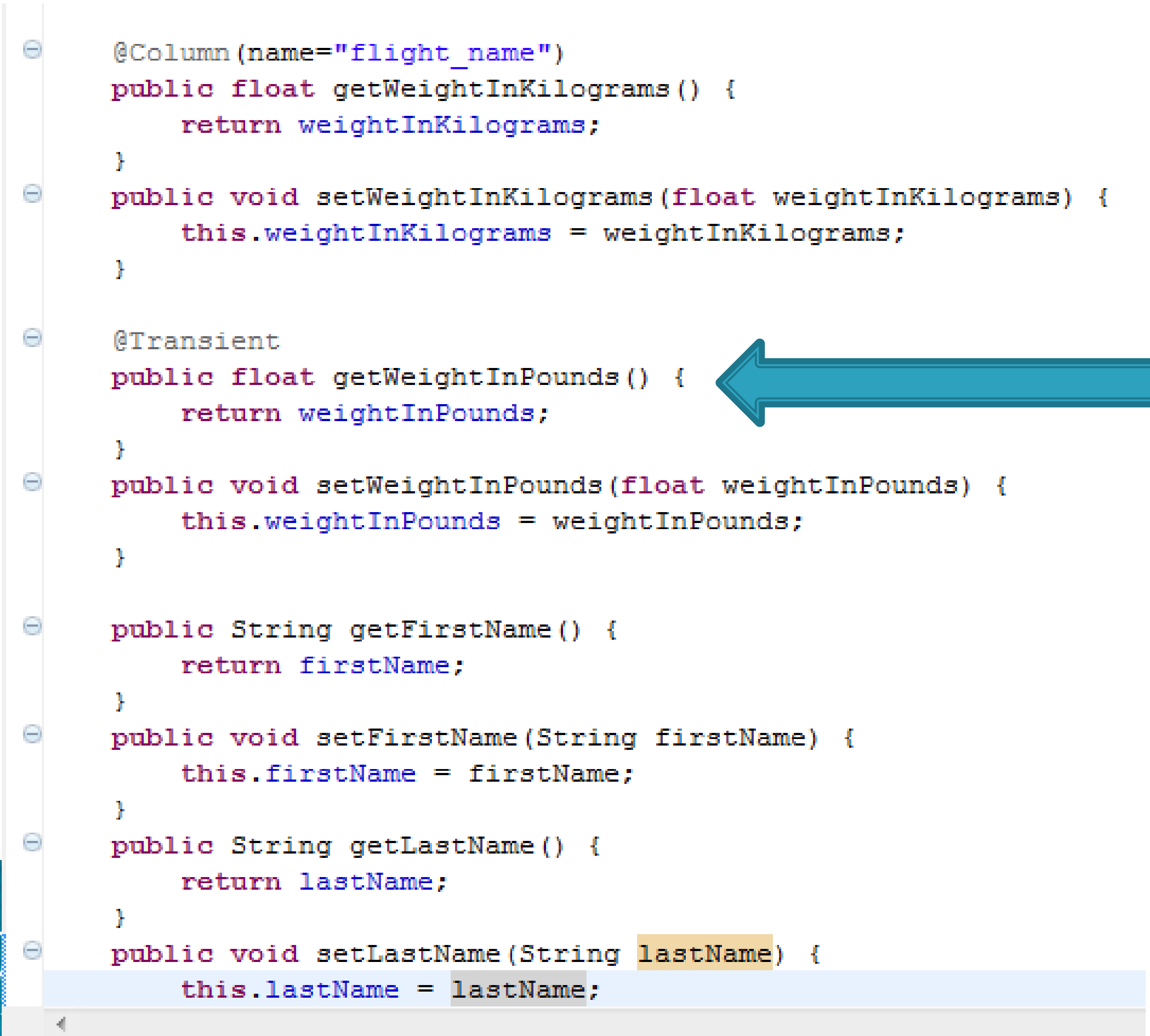
Transient

- Transient properties will be ignored (not saved to the database)



```
@Transient
public float getWeightInPounds() {
    return weightInPounds;
}
public void setWeightInPounds(float weightInPounds) {
    this.weightInPounds = weightInPounds;
}
```

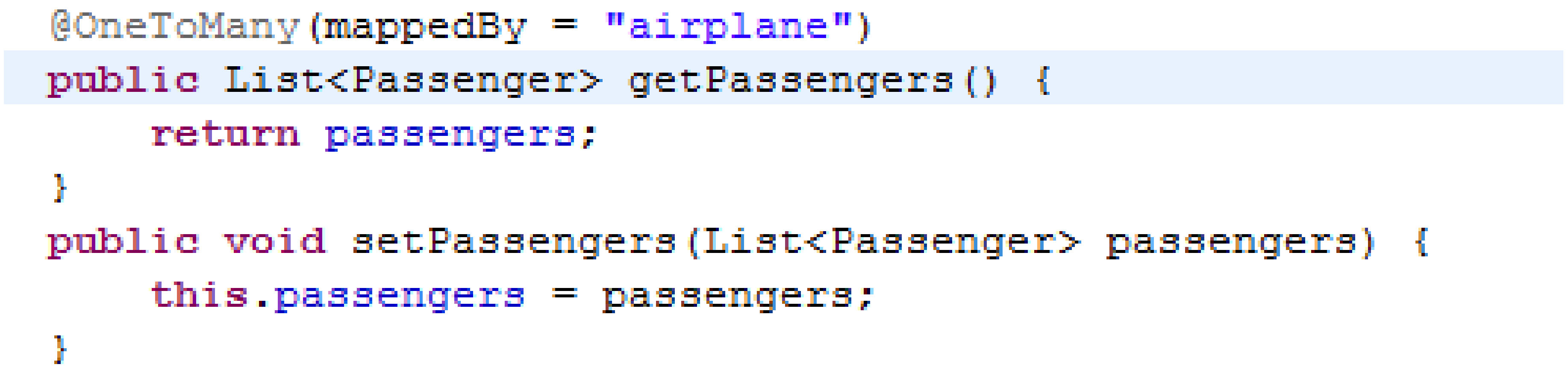
Properties are persisted by default!



```
    @Column(name="flight_name")
    public float getWeightInKilograms() {
        return weightInKilograms;
    }
    public void setWeightInKilograms(float weightInKilograms) {
        this.weightInKilograms = weightInKilograms;
    }

    @Transient
    public float getWeightInPounds() {
        return weightInPounds;
    }
    public void setWeightInPounds(float weightInPounds) {
        this.weightInPounds = weightInPounds;
    }

    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
```

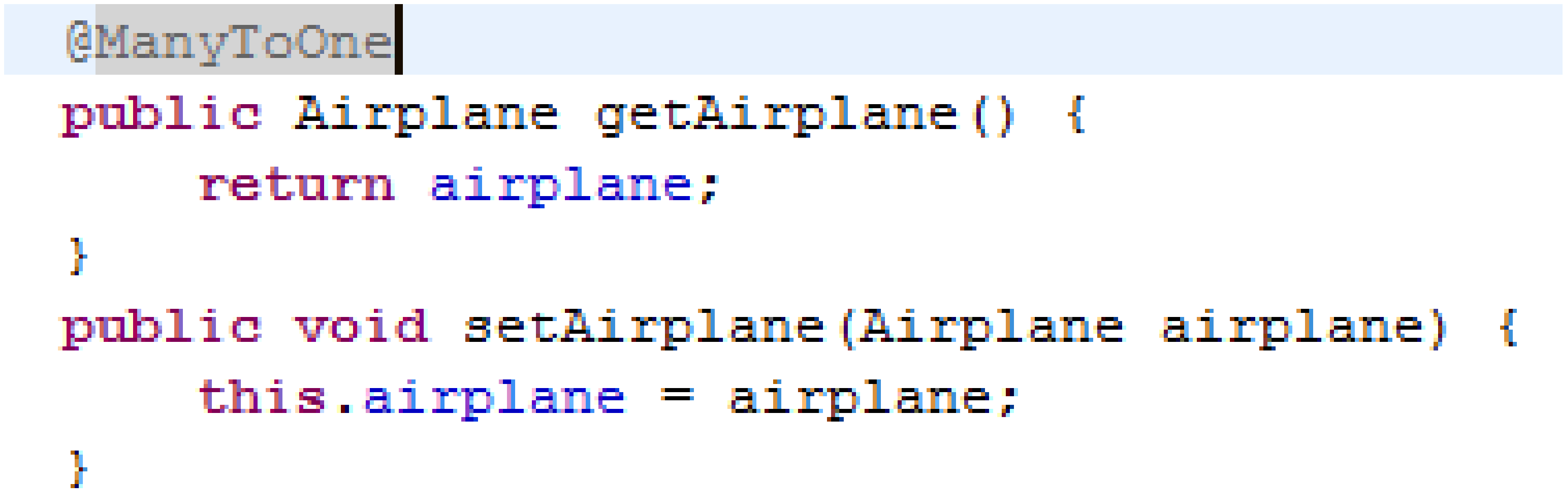
Only property not saved to the database!

Associations (Airplane.java)



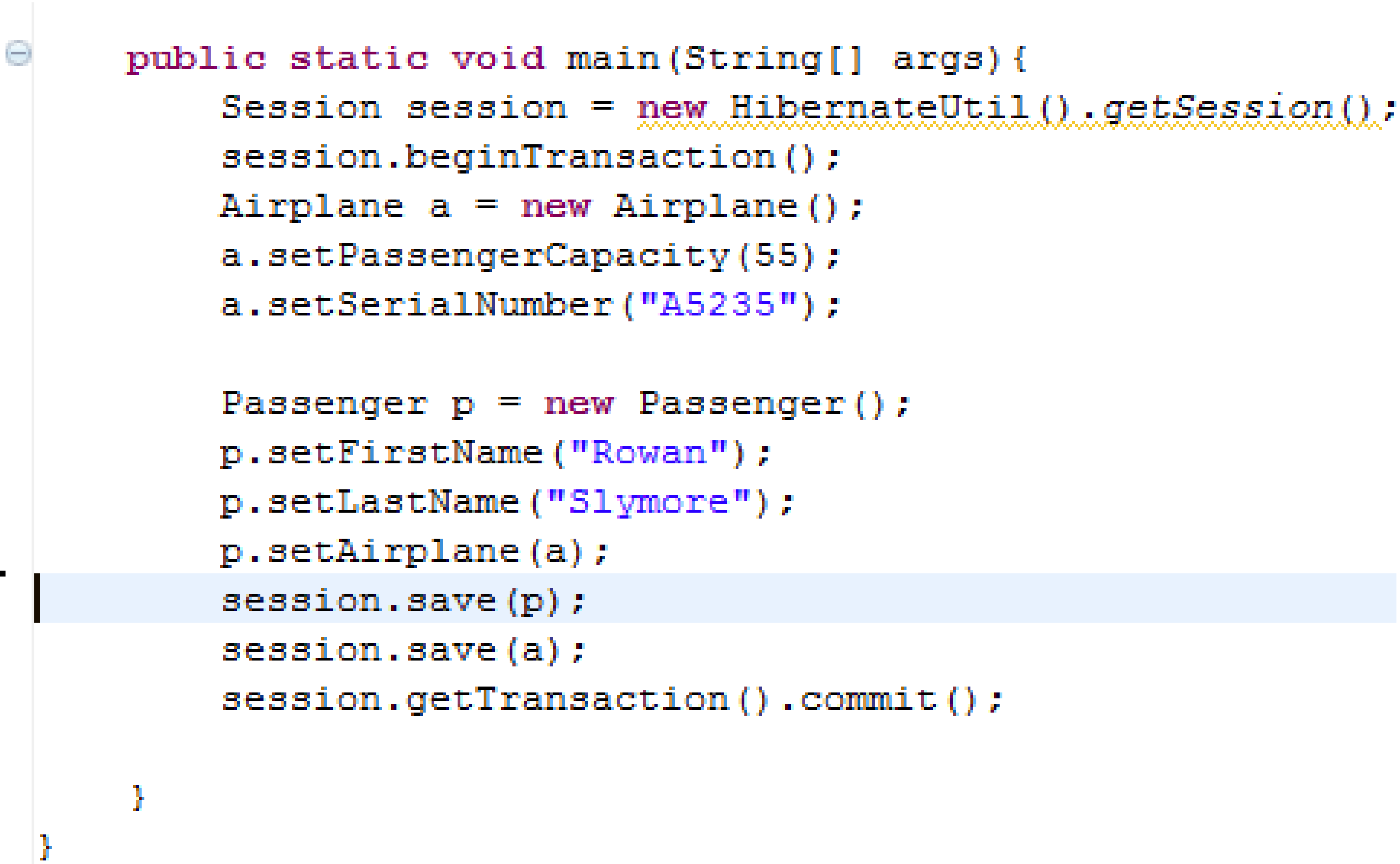
```
@OneToMany(mappedBy = "airplane")
public List<Passenger> getPassengers() {
    return passengers;
}
public void setPassengers(List<Passenger> passengers) {
    this.passengers = passengers;
}
```

Associations (Passenger.java)



```java
@ManyToOne
public Airplane getAirplane() {
     return airplane;
}
public void setAirplane(Airplane airplane) {
     this.airplane = airplane;
}
```

Working Associations



```
public static void main(String[] args){
    Session session = new HibernateUtil().getSession();
    session.beginTransaction();
    Airplane a = new Airplane();
    a.setPassengerCapacity(55);
    a.setSerialNumber("A5235");

    Passenger p = new Passenger();
    p.setFirstName("Rowan");
    p.setLastName("Slymore");
    p.setAirplane(a);
    session.save(p);
    session.save(a);
    session.getTransaction().commit();


}
}
```