**BD**

**LISTA 10**

**WIOLETTA ŁUPKOWSKA, 244831**

**PONIEDZIAŁEK, 13:15**

**1.1**

*<?***xml version="1.0" encoding="UTF-8"***?>*<**project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
  
 <**groupId**>edu.ib</**groupId**>  
 <**artifactId**>MySQL\_Hibernate\_Demo\_Maven</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
  
 <**dependencies**>  
  
 *<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->* <**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 <**version**>8.0.18</**version**>  
 </**dependency**>  
  
 *<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager -->* <**dependency**>  
 <**groupId**>org.hibernate</**groupId**>  
 <**artifactId**>hibernate-entitymanager</**artifactId**>  
 <**version**>5.4.9.Final</**version**>  
 </**dependency**>  
  
  
 </**dependencies**>  
  
  
 <**build**>  
 <**plugins**>  
 <**plugin**>  
 <**artifactId**>maven-compiler-plugin</**artifactId**>  
 <**version**>3.8.1</**version**>  
 <**configuration**>  
 <**source**>10</**source**>  
 <**target**>10</**target**>  
 </**configuration**>  
  
 </**plugin**>  
 </**plugins**>  
 </**build**>  
  
</**project**>

*<?***xml version = "1.0" encoding = "utf-8"***?>***<!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd"*>***<**hibernate-configuration**>  
 <**session-factory**>  
  
 <**property name = "hibernate.dialect"**>org.hibernate.dialect.MySQL8Dialect</**property**>  
  
 <**property name = "hibernate.connection.driver\_class"**>com.mysql.jdbc.Driver</**property**>  
  
 <**property name = "hibernate.connection.url"**>jdbc:mysql://127.0.0.1:3306/nagrody?useTimezone=true**&amp;**serverTimezone=CET</**property**>  
  
 <**property name = "hibernate.connection.username"**>Wiola</**property**>  
  
 <**property name = "hibernate.connection.password"**>.</**property**>  
  
 *<!-- List of XML mapping files -->* <**mapping resource = "Laureaci.hbm.xml"**/>  
  
 <**mapping class="Laureaci"**/>  
  
  
 </**session-factory**>  
</**hibernate-configuration**>

*<?***xml version='1.0' encoding='UTF-8'***?>***<!DOCTYPE hibernate-mapping PUBLIC  
 "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  
 "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd"*>***<**hibernate-mapping**>  
 <**class name="Laureaci" table="laureaci"**>  
 <**id name="id" type="int" column="id"**>  
 <**generator class="assigned"**></**generator**> *<!-- automatyczna generacja klucza glownego -->* </**id**>  
  
 <**property name="rok" column="rok"**></**property**> *<!-- mapowanie atrybutu z kolumna w bazie -->* <**property name="dane" column="dane"**></**property**>  
 <**property name="wiek" column="wiek"**></**property**>  
 <**property name="nazwa\_filmu" column="nazwa\_filmu"**></**property**>  
 <**property name="plec" column="plec"**></**property**>  
  
 </**class**>

</**hibernate-mapping**>

**import** javax.persistence.\*;  
**import** java.util.\*;  
  
@Entity  
@Table(name = **"laureaci"**)  
**public class** Laureaci {  
  
 @Id  
 @GeneratedValue*//(strategy=GenerationType.IDENTITY)* @Column(name = **"id"**)  
 **private int id**;  
 @Column(name = **"rok"**)  
 **private int rok**;  
 @Column(name = **"dane"**)  
 **private** String **dane**;  
 @Column(name = **"wiek"**)  
 **private int wiek**;  
 @Column(name = **"nazwa\_filmu"**)  
 **private** String **nazwa\_filmu**;  
 @Column(name = **"plec"**)  
 **private** String **plec**;  
  
  
 **public** Laureaci() {  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 **public void** setId(**int** id) {  
 **this**.**id** = id;  
 }  
  
 **public int** getRok() {  
 **return rok**;  
 }  
  
 **public void** setRok(**int** rok) {  
 **this**.**rok** = rok;  
 }  
  
 **public** String getDane() {  
 **return dane**;  
 }  
  
 **public void** setDane(String dane) {  
 **this**.**dane** = dane;  
 }  
  
 **public int** getWiek() {  
 **return wiek**;  
 }  
  
 **public void** setWiek(**int** wiek) {  
 **this**.**wiek** = wiek;  
 }  
  
 **public** String getNazwa\_filmu() {  
 **return nazwa\_filmu**;  
 }  
  
 **public void** setNazwa\_filmu(String nazwa\_filmu) {  
 **this**.**nazwa\_filmu** = nazwa\_filmu;  
 }  
  
 **public** String getPlec() {  
 **return plec**;  
 }  
  
 **public void** setPlec(String plec) {  
 **this**.**plec** = plec;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Laureaci{"** +  
 **"id="** + **id** +  
 **", rok="** + **rok** +  
 **", dane='"** + **dane** + **'\''** +  
 **", wiek="** + **wiek** +  
 **", nazwa\_filmu='"** + **nazwa\_filmu** + **'\''** +  
 **", plec='"** + **plec** + **'\''** +  
 **'}'**;  
 }  
}

**import** org.hibernate.Session;  
**import** org.hibernate.SessionFactory;  
**import** org.hibernate.Transaction;  
**import** org.hibernate.cfg.Configuration;  
**import** org.hibernate.query.Query;  
  
  
**import** java.util.List;  
  
**public class** Test {  
  
 **private static** SessionFactory *factory*;  
  
 **public static void** main(String[] args) {  
  
 **try** { *//czy w ogole uda nam sie zbudowac konfiguracje  
 factory* = **new** Configuration().configure().buildSessionFactory();  
 } **catch** (Throwable ex) {  
 System.***err***.println(**"Failed to create sessionFactory object."** + ex);  
 **throw new** ExceptionInInitializerError(ex);  
 }  
  
 Session session = *factory*.openSession();  
 Transaction tx = session.beginTransaction();  
  
 System.***out***.println(**"\*Wyświetl średni wiek zwycięzcy i zwyciężczyni (osobno)"**);  
 *showQueryResults*(**"select avg(wiek) as wiekF from Laureaci where plec='F'"**);  
 *showQueryResults*(**"select avg(wiek) as wiekF from Laureaci where plec='M'"**);  
  
 System.***out***.println(**"\*Wyświetl, ile nagród dostali razem członkowie rodziny Fonda"**);  
 *showQueryResults*(**"select count(dane) from Laureaci where dane like '%Fonda%'"**);  
  
  
 System.***out***.println(**"\*Wyświetl wiek najmłodszego i najstarszego zwycięzcy oraz najmłodszej i najstarszej"** +  
 **"zwyciężczyni (osobno)."**);  
  
 *//showQueryResults("select wiek from Laureaci where plec='F' order by wiek asc limit 1 ");* List najstarsza= session.createSQLQuery(**"select wiek from Laureaci where plec='F' order by wiek asc"**).list();  
 System.***out***.println(**"Wiek najstarszej: "** + najstarsza.get(najstarsza.size()-1));  
  
 List najmlodsza= session.createSQLQuery(**"select wiek from Laureaci where plec='F' order by wiek desc"**).list();  
 System.***out***.println(**"Wiek najmlodszej: "**+najmlodsza.get(najmlodsza.size()-1));  
  
 List najstarszy= session.createSQLQuery(**"select wiek from Laureaci where plec='M' order by wiek asc"**).list();  
 System.***out***.println(**"Wiek najstarszego: "**+najstarszy.get(najmlodsza.size()-1));  
  
 List najmlodszy= session.createSQLQuery(**"select wiek from Laureaci where plec='M' order by wiek desc"**).list();  
 System.***out***.println(**"Wiek najmlodszego: "**+najmlodszy.get(najmlodsza.size()-1));  
  
 System.***out***.println(**"\*Oblicz, ile procent nagród zdobyły kobiety."**); *showQueryResults*(**" select ((select count(plec) from Laureaci where plec='F')\*100/((select count(plec) from Laureaci where plec='F')+(select count(plec) from Laureaci where plec='M')))"**);  
System.***out***.println(**" Wyświetl imiona i nazwiska aktorów/aktorek, którzy zdobyli więcej niż jednego Oskara\n"** +  
 **"i uszereguj ich wg malejącej liczby nagród."**);  
 *showQueryResults*(**"SELECT new list(dane ,COUNT(dane) )FROM Laureaci GROUP BY dane having COUNT(dane)>1 ORDER BY COUNT(dane) desc "**);  
  
 System.***out***.println(**" W pierwszych latach Oskary były przyznawane więcej niż jednej osobie w danej kategorii.\n"** +  
 **"Wyznacz, ile było rozdań nagród."**);  
 *showQueryResults*(**"SELECT new list( rok ,COUNT(rok)) FROM Laureaci GROUP BY rok ORDER BY COUNT(rok)"**);  
  
 System.***out***.println(**" Wybierz trzech aktorów z listy stu najwybitniejszych aktorów wg serwisu IMDb2\n"** +  
 **"i wyświetl\n"** +  
 **"dla nich zestawienie w formie: imię i nazwisko, rok nagrody i film. "**);  *showQueryResults*(**"select new list(dane, rok, nazwa\_filmu) from Laureaci where dane ='Al Pacino' or dane ='Jack Nicholson' or dane='Anthony Hopkins'"**);  
  
 System.***out***.println(**" Wyznacz, ile było różnych laureatów pośród kobiet i mężczyzn (osobno)."**);  
 *showQueryResults*(**"SELECT count(distinct dane) FROM Laureaci where plec='F'"**);  
 *showQueryResults*(**"SELECT count(distinct dane) FROM Laureaci where plec='M'"**);  
   
 session.getTransaction().commit();  
 session.close();  
  
  
 }  
  
  
 **public static void** showQueryResults(String queryString) {  
  
 Session session = *factory*.openSession();  
 Query query = session.createQuery(queryString);  
  
 List list = query.list();  
  
 **for** (**int** i = 0; i < list.size(); i++)  
 System.***out***.println(list.get(i));  
  
 }

}

1.2











