МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра ВТ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Объектно-ориентированное программирование»

ТЕМА: Hibernate и подключение к БД

Студент гр. 9308 Семенов А.И.

Преподаватель Гречухин М. Н.

Санкт-Петербург

2021

Содержание

Введение

Цель работы: разработать базовые классы для последующей разработки программного комплекса (ПК) на объектно-ориентированном языке Java, научиться создавать UML-диаграммы для поставленной задачи.

Описание ПК:

ПК для менеджера соревнований автогонщиков. В ПК должны храниться сведения о командах, гонщиках и трассах. Менеджер соревнований может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- состав гонщиков команды;
- призеры гонок по каждой трассе;
- график проведения соревнований;
- количество очков, набранных гонщиками.

1. База данных

Для поставленной задачи была создана следующая база данных:

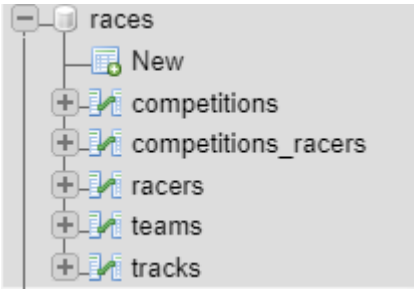

Рисунок 1. База данных

Где races – название базы.

Таблицы выглядят следующим образом:

id	name	date	track_id	winner_id

Рисунок 2. Таблица competitions

Competition_id	racers_id

Рисунок 3. Таблица competitions_racers

id	name	surname	score	teamid

Рисунок 4. Таблица racers

id	name

Рисунок 5. Таблица teams

id	name	country

Рисунок 6. Таблица tracks

2. Разметка классов

Класс Racer (гонщик)

```
package Races;

import javax.persistence.*;

@Entity
@Table(name="racers")
public class Racer
{
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="name")
    private String name;

    @Column(name="surname")
    private String surname;

    @ManyToOne (optional=false)
    @JoinColumn (name="teamid")
    private Team team;

    @Column(name="score")
    private int score = 0;

    public Racer() {}

    public Racer(String _name, String _surname, Team _team)
    {
        name = _name;
        surname = _surname;
        team = _team;
    }

    public int getId()
    {
        return id;
    }

    public String getName()
    {
        return name;
    }

    public void setName(String newName)
    {
        name = newName;
    }

    public String getSurname()
    {
        return surname;
    }

    public void setSurname(String newSurname)
    {
        surname = newSurname;
    }
```

```
    public Team getTeam()
    {
        return team;
    }

    public boolean setTeam(Team newTeam)
    {
        if(newTeam != null)
        {
            team = newTeam;
            return true;
        }
        return false;
    }

    public int getScore()
    {
        return score;
    }

    public void setScore(int newScore)
    {
        score = newScore;
    }

    @Override
    public String toString()
    {
        return name + " " + surname + ", score: " + score;
    }
}
```

Класс Team (команда)

```
package Races;

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

@Entity
@Table(name="teams")
public class Team
{
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="name")
    private String name;

    @OneToMany (mappedBy="team", fetch = FetchType.EAGER)
    private List<Racer>racers;

    public Team() {}

    public Team(String t_name, List<Racer> t_racers)
    {
        name = t_name;
        racers = t_racers;
```

```
}

public Team(String t_name)
{
    name = t_name;
    racers = new ArrayList<Racer>();
}

public int getId()
{
    return id;
}

public String getName()
{
    return name;
}

public void setName(String newName)
{
    name = newName;
}

public ArrayList<Racer> getRacers()
{
    return new ArrayList<Racer>(racers);
}

public boolean setRacers(ArrayList<Racer> newRacers)
{
    if(newRacers != null)
    {
        racers = newRacers;
        return true;
    }
    return false;
}

public boolean addRacer(Racer racer)
{
    if(racer != null)
    {
        racers.add(racer);
        racer.setTeam(this);
        return true;
    }
    return false;
}

public boolean removeRacer(Racer racer)
{
    int index = racers.indexOf(racer);
    if(index != -1)
    {
        racers.remove(index);
        return true;
    }
    return false;
}

public boolean isEquals(Team other)
{
    return this.name == other.name;
}
```

```
    @Override
    public boolean equals(Object obj)
    {
        if (obj == this) return true;
        if (obj == null) return false;
        if( this.getClass() != obj.getClass() ) return false;
        Team other = (Team)obj;
        return this.isEquals(other);
    }

    @Override
    public String toString()
    {
        int score = 0;
        for(int i = 0; i < racers.size(); i++)
            score += racers.get(i).getScore();
        return "Team: " + name + ". Total score: " + score + ". Number of
racers in team: " + racers.size();
    }
}
```

Класс Track (трасса)

```
package Races;

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

@Entity
@Table(name="tracks")
public class Track
{
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="name")
    private String name;

    @Column(name="country")
    private String country;

    public Track() {}

    public Track(String t_name, String t_country)
    {
        name = t_name;
        country = t_country;
    }

    public int getId()
    {
        return id;
    }

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

```
    }

    public void setName(String newName)
    {
        name = newName;
    }

    public String getCountry()
    {
        return country;
    }

    public void setCountry(String newCountry)
    {
        country = newCountry;
    }

    @Override
    public String toString()
    {
        return name + "(" + country + ")";
    }
}
```

Класс Competition (соревнование)

```
package Races;

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

@Entity
@Table(name="competitions")
public class Competition
{
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="name")
    private String name;

    @Column(name="date")
    private Date date;

    @OneToOne (optional=false)
    private Track track;

    @OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL,
orphanRemoval = true)
    private List<Racer>racers;

    @OneToOne (optional=false)
    private Racer winner;

    public Competition() {}

    public Competition(String _name, Date _date, Track _track, List<Racer>
_racers)
```

```
{
    name = _name;
    date = _date;
    track = _track;
    racers = _racers;
}

public Competition(String _name, Date _date, Track _track)
{
    name = _name;
    date = _date;
    track = _track;
    racers = new ArrayList<Racer>();
}

public int getId()
{
    return id;
}

public String getName()
{
    return name;
}

public void setName(String newName)
{
    name = newName;
}

public Date getDate()
{
    return date;
}

public boolean setDate(Date _date)
{
    if(_date != null)
    {
        date = _date;
        return true;
    }
    return false;
}

public Track getTrack()
{
    return track;
}

public boolean setTrack(Track newTrack)
{
    if(newTrack != null)
    {
        track = newTrack;
        return true;
    }
    return false;
}

public ArrayList<Racer> getRacers()
{
    return new ArrayList<Racer>(racers);
}
```

```
    public boolean setRacers(ArrayList<Racer> newRacers)
    {
        if(newRacers != null)
        {
            racers = newRacers;
            return true;
        }
        return false;
    }

    public boolean addRacer(Racer newRacer)
    {
        if(newRacer != null)
        {
            racers.add(newRacer);
            return true;
        }
        return false;
    }

    public boolean removeRacer(Racer racer)
    {
        if(racer != null)
        {
            racers.remove(racer);
            return true;
        }
        return false;
    }

    public Racer getWinner()
    {
        return winner;
    }

    public boolean setWinner(Racer newWinner)
    {
        if(newWinner != null)
        {
            winner = newWinner;
            return true;
        }
        return false;
    }

    @Override
    public String toString()
    {
        String out = "Competition " + name + ". Track: " + track + ". Date: "
+ date;
        return out;
    }
}
```

Вывод

При выполнении курсового проектирования реализовано подключение к БД, сохранение в БД и получение данных из нее. Приобретены практические навыки по проектированию и разработке программного обеспечения на объектно-ориентированном языке Java.