

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра вычислительной техники

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Объектно-ориентированное программирование»
Тема: Hibernate и подключение к MySQL

Студент гр. 9308

Яловега Н. В.

Преподаватель

Гречухин М.Н.

Санкт-Петербург

2021

Цель работы

Целью работы является ознакомление с технологиями Hibernate и JPA, а также Базами Данных MySQL.

Уточнение задания

Создать Базу Данных с помощью утилиты MySQL WorkBench. Создать проект Maven и указать зависимости MySQL, Hibernate и JPA. Аннотация имеющегося прототипа проекта к сущностям для записи в Базу Данных.

Ход работы

1. База Данных.

Получившаяся структура таблиц в Базе Данных(с настройками и зависимостями):

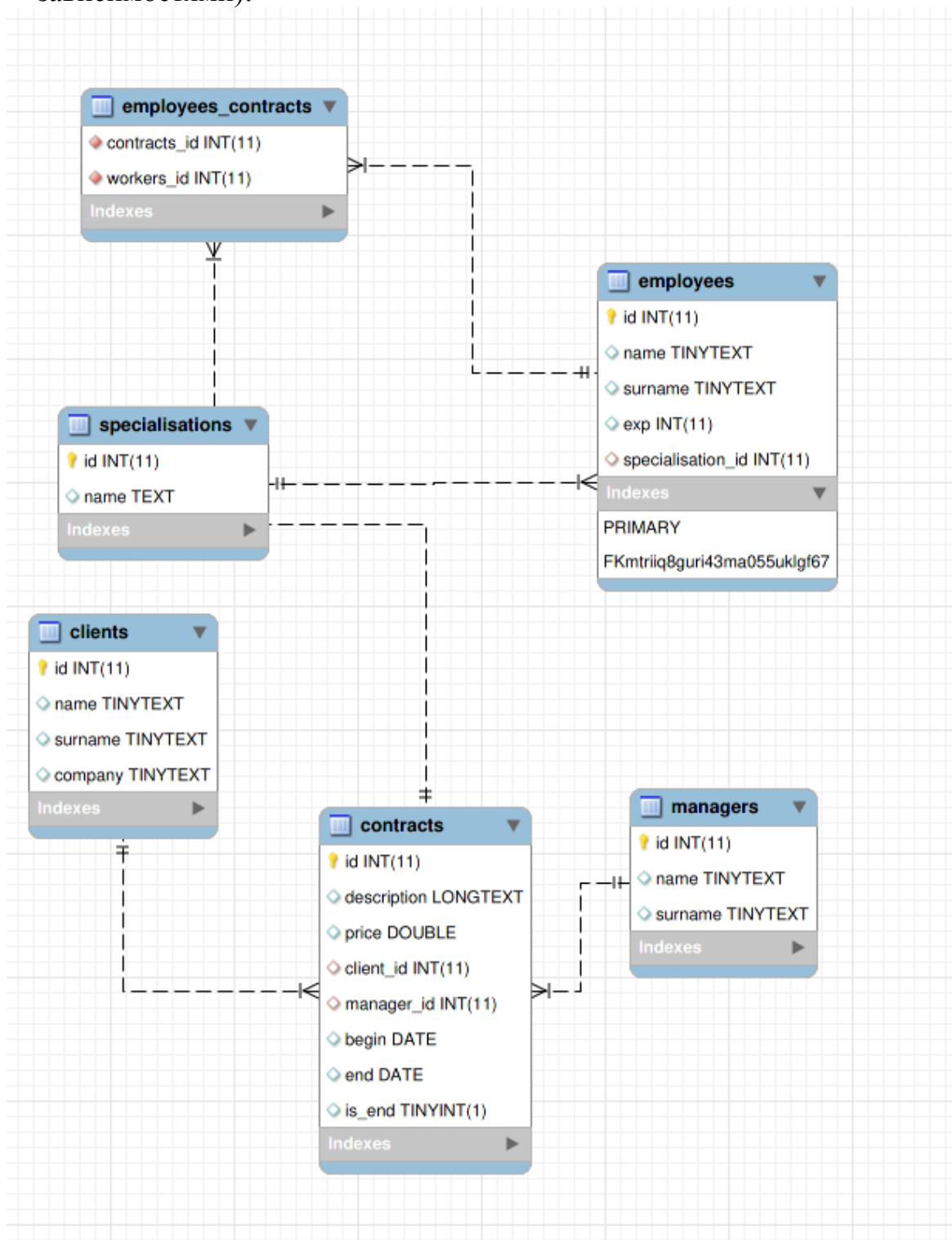


Рис.1. Структура Базы Данных

2. Аннотации

Аннотации для класса Person

```
@MappedSuperclass
public class Person
{
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    protected int id;

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

    @Column(name="surname")
    protected String surname;
}
```

Аннотации для класса Specialisation

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

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

    @OneToMany(mappedBy = "specialisation", cascade =
CascadeType.ALL, orphanRemoval = true)
    private List<Employee> employees;
}
```

Аннотации для класса Employee

```
@Entity(name = "employees")
@Table(name = "employees")
public class Employee extends Person
{
    @Column(name="exp")
    private int exp;

    @ManyToOne
    @JoinColumn (name="specialisation_id")
    private Specialisation specialisation;

    @ManyToMany(fetch = FetchType.EAGER, cascade =
CascadeType.ALL)
}
```

```

        @JoinTable(
            name = "employees_contracts",
            joinColumns = { @JoinColumn(name = "workers_id",
referencedColumnName = "id" )},
            inverseJoinColumns = { @JoinColumn(name =
"contracts_id", referencedColumnName = "id") }
        )
        private List<Contract> contracts;

```

Аннотации для класса Manager

```

@Entity(name = "managers")
@Table(name = "managers")
public class Manager extends Person
{

    @OneToMany(mappedBy = "manager")
    private List<Contract> contracts;

```

Аннотации для класса Client

```

@Entity(name = "clients")
@Table(name = "clients")
public class Client extends Person
{

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

    @OneToMany(mappedBy = "client")
    private List<Contract> client_contracts;

```

Аннотации для класса Contract

```

@Entity(name = "contracts")
@Table(name="contracts")
public class Contract
{

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

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

    @Column(name="price")
    private double price;

    @ManyToOne
    @JoinColumn (name = "client_id")
    private Client client;

```

```
@ManyToOne
@JoinColumn (name = "manager_id")
private Manager manager;

@ManyToMany(mappedBy = "contracts", fetch = FetchType.EAGER)
private List<Employee> workers;

@Column(name="begin")
private Date dateBegin;

@Column(name="end")
private Date dateEnd;

@Column(name="is_end")
private boolean isEnd;
```

Выводы.

При выполнении лабораторной работы был расширен прототип конечного проекта для работы с БД. Были приобретены практические навыки по проектированию БД в MySQL Workbench, работе с Hibernate и Java Persistence API(JPA) и разработке программного обеспечения на объектно-ориентированном языке Java.

Приложение А (Исходный код)

Specialisation.java

```
package Factory.model;

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

/**
 * Класс специализации работника
 * @author Яловега Никита 9308
 * @version 0.1
 */
@Entity(name = "specialisations")
@Table(name = "specialisations")
public class Specialisation
{

    /** Уникальный идентификатор профессии */
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    /** Название профессии */
    @Column(name="name")
    private String name;

    /** Рабочие данной профессии */
    @OneToMany(mappedBy = "specialisation", cascade = CascadeType.ALL,
    orphanRemoval = true)
    private List<Employee> employees;

    public Specialisation() {}

    public Specialisation(String name)
    {
        this.name = name;
    }

    /**
     * Метод получения значения поля {@link Specialisation#id}
     * @return возвращает уникальный идентификатор
     */
}
```



```

    */
    public int getID()
    {
        return id;
    }

    /**
     * Метод определения значения поля {@link Specialisation#id}
     * @param newID - новый идентификатор
     */
    public void setID(int newID)
    {
        id = newID;
    }

    /**
     * Метод получения значения поля {@link Specialisation#name}
     * @return возвращает название специализации
     */
    public String getName()
    {
        return name;
    }

    /**
     * Метод определения значения поля {@link Specialisation#name}
     * @param newName - новое название специализации
     */
    public void setName(String newName)
    {
        name = newName;
    }

    /**
     * Метод добавления новых работников {@link Specialisation#employees}
     * @param newEmployee - новый работник
     */
    public void addEmployee(Employee newEmployee)
    {
        employees.add(newEmployee);
        // связываем сотрудника с этим отделом
        newEmployee.setSpecialisation(this);
    }

```

```

/**
 * Метод получения значения поля {@link Specialisation#employees}
 * @return Работники данной профессии
 */
public List<Employee> getEmployees()
{
    return employees;
}

/**
 * Метод удаления работников из профессии {@link
Specialisation#employees}
 * @param e - работник, которого нужно убрать
 */
public void removeEmployee(Employee e)
{
    employees.remove(e);
}

/**
 * Функция получения всей информации об объекте
 * @return - массив строк с данными
 */
public String[] toTableFormat()
{
    return new String[] {String.valueOf(getID()), getName()};
}
}

```

Person.java

```

package Factory.model;

import javax.persistence.*;

/**
 * Класс человека
 * @author Яловега Никита 9308
 * @version 0.1
 */
@Entity
public class Person
{
    /** Поле уникального идентификатора */

```

```

@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)
@Column(name="id")
protected int id;

/** Поле имени человека */
@Column(name="name")
protected String name;

/** Поле фамилии человека */
@Column(name="surname")
protected String surname;

/** Стандартный конструктор человека */
public Person() {}

/**
 * Конструктор - создание нового объекта Person
 * @param name - имя
 * @param surname - фамилия
 */
public Person(String name, String surname)
{
    this.name = name;
    this.surname = surname;
}

/**
 * Функция получения значения поля {@link Person#id}
 * @return возвращает уникальный идентификатор человека
 */
public int getId()
{
    return id;
}

/**
 * Функция определения значения поля {@link Person#id}
 * @param newID - новый идентификатор пользователя
 */
public void setId(int newID)
{
    id = newID;
}

```

```

/**
 * Функция получения значения поля {@link Person#name}
 * @return возвращает имя человека
 */
public String getName()
{
    return name;
}

/**
 * Процедура определения значения поля {@link Person#name}
 * @param newName - новое имя человека
 */
public void setName(String newName)
{
    name = newName;
}

/**
 * Функция получения значения поля {@link Person#surname}
 * @return возвращает фамилию человека
 */
public String getSurname()
{
    return surname;
}

/**
 * Процедура определения значения поля {@link Person#name}
 * @param newSurname - новая фамилия человека
 */
public void setSurname(String newSurname)
{
    surname = newSurname;
}
}

```

Manager.java

```

package Factory.model;

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

```

```

/**
 * Класс менеджера завода по производству металлических изделий.
 * Наследник класса {@link Person}
 * @author Яловега Никита 9308
 * @version 0.1
 */
@Entity(name = "managers")
@Table(name = "managers")
public class Manager extends Person
{
    /** Контракты, в которые подписывал менеджер */
    @OneToMany(mappedBy = "manager")
    private List<Contract> contracts;

    public Manager() {}

    /**
     * Конструктор - создание нового объекта Manager
     * @param name - имя
     * @param lastName - фамилия
     */
    public Manager(String name, String lastName)
    {
        super(name, lastName);
    }

    /**
     * Процедура добавления новых контрактов рабочему.
     * @param newContract - новый контракт, который выполняет рабочий
     */
    public void addContract(Contract newContract)
    {
        contracts.add(newContract);
        newContract.setManager(this); // добавляем в контракт рабочего
    }

    /**
     * Функция получения значения поля {@link Manager#contracts}
     * @return возвращает контракты, который выполняет рабочий
     */
    public List<Contract> getContracts()
    {
        return contracts;
    }
}

```

```

    }

    /**
     * Процедура удаления контрактов, которые выполняет рабочий
     * @param c - контракт
     */
    public void removeContract(Contract c)
    {
        contracts.remove(c);
        c.setManager(null);
    }

    /**
     * Процедура установки контрактов, которые выполняет рабочий
     * @param c - список контракт
     */
    public void setContract(List<Contract> c)
    {
        contracts = c;
    }

    /**
     * Функция получения всей информации об объекте
     * @return - массив строк с данными
     */
    public String[] toTableFormat()
    {
        return new String[] {String.valueOf(id), name, surname};
    }
}

```

Employee.java

```

package Factory.model;

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

/**
 * Класс сотрудника завода по производству металлических изделий.
 * Наследник класса {@link Person}
 * @author Яловега Никита 9308
 * @version 0.1
 */

```

```

@Entity(name = "employees")
@Table(name = "employees")
public class Employee extends Person
{
    /** Поле опыта работы */
    @Column(name="exp")
    private int exp;

    /** Поле профессии рабочего */
    @ManyToOne
    @JoinColumn (name="specialisation_id")
    private Specialisation specialisation;

    /** Контракты, в которых участвует рабочий */
    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinTable(
        name = "employees_contracts",
        joinColumns = { @JoinColumn(name = "workers_id",
referencedColumnName = "id" )},
        inverseJoinColumns = { @JoinColumn(name = "contracts_id",
referencedColumnName = "id" ) }
    )
    private List<Contract> contracts;

    /**
     * Стандартный конструктор
     */
    public Employee() {}

    /**
     * Конструктор - создание нового объекта Employee
     * @param name - имя
     * @param lastName - фамилия
     * @param exp - опыт работы
     * @param specialisation - профессия
     */
    public Employee(String name, String lastName, int exp, Specialisation
specialisation)
    {
        super(name, lastName);
        this.exp = exp;
        this.specialisation = specialisation;
        this.contracts = null;
    }
}

```

```

/**
 * Функция получения значения поля {@link Employee#exp}
 * @return возвращает опыт работы рабочего
 */
public int getExp()
{
    return exp;
}

/**
 * Процедура определения значения поля {@link Employee#exp}
 * @param newExp - новая фамилия человека
 */
public void setExp(int newExp)
{
    exp = newExp;
}

/**
 * Процедура определения значения поля {@link Employee#specialisation}
 * @param d - профессия сотрудника
 */
public void setSpecialisation(Specialisation d)
{
    specialisation = d;
}

/**
 * Функция получения значения поля {@link Employee#specialisation}
 * @return возвращает профессию сотрудника
 */
public Specialisation getSpecialisation()
{
    return specialisation;
}

/**
 * Процедура добавления новых контрактов рабочему.
 * @param newContract - новый контракт, который выполняет рабочий
 */
public void addContract(Contract newContract)
{
    contracts.add(newContract);
}

```



```

        newContract.addWorker(this); // добавляем в контракт рабочего
    }

    /**
     * Функция получения значения поля {@link Employee#contracts}
     * @return возвращает контракты, который выполняет рабочий
     */
    public List<Contract> getContracts()
    {
        return contracts;
    }

    /**
     * Процедура удаления контрактов, которые выполняет рабочий
     * @param c - контракт
     */
    public void removeContract(Contract c)
    {
        contracts.remove(c);
        c.removeWorker(this);
    }

    /**
     * Функция получения всей информации об объекте
     * @return - массив строк с данными
     */
    public String[] toTableFormat()
    {
        return new String[] {String.valueOf(id), name, surname, String.valueOf(exp),
specialisation.getName()};
    }
}

```

Client.java

```

package Factory.model;

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

/**
 * Класс клиента завода по производству металлических изделий.
 * Наследник класса {@link Person}

```

```

* @author Яловега Никита 9308
* @version 0.1
*/
@Entity(name = "clients")
@Table(name = "clients")
public class Client extends Person
{
    /** Название компании клиента */
    @Column(name="company")
    private String company;

    /** Контракты клиента */
    @OneToMany(mappedBy = "client")
    private List<Contract> client_contracts;

    public Client() { }

    /**
     * Конструктор - создание нового объекта {@link Client}
     * @param name - имя
     * @param lastName - фамилия
     * @param company - компания
     */
    public Client(String name, String lastName, String company)
    {
        super(name, lastName);
        this.company = company;
    }

    /**
     * Функция получения значения поля {@link Client#company}
     * @return возвращает название компании клиента
     */
    public String getCompany()
    {
        return company;
    }

    /**
     * Процедура определения значения поля {@link Client#company}
     * @param newCompany - новая название компании клиента
     */
    public void setCompany(String newCompany)
    {

```

```

    company = newCompany;
}

/**
 * Процедура добавления новых контрактов клиента
 * @param newContract - новый контракт
 */
public void addContract(Contract newContract)
{
    client_contracts.add(newContract);
    newContract.setClient(this); // связываем контракт с этим клиентом
}

/**
 * Функция получения значения поля {@link Client#client_contracts}
 * @return возвращает контракты клиента
 */
public List<Contract> getContracts()
{
    return client_contracts;
}

/**
 * Процедура удаления контрактов
 * @param c - контракт
 */
public void removeContract(Contract c)
{
    client_contracts.remove(c);
    c.setClient(null);
}

/**
 * Функция получения всей информации об объекте
 * @return - массив строк с данными
 */
public String[] toTableFormat()
{
    return new String[] {String.valueOf(id), name, surname, company};
}
}

```

Contract.java

```
package Factory.model;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import javax.persistence.*;

/**
 * Класс контракта завода по производству металлических изделий.
 * @author Яловега Никита 9308
 * @version 0.1
 */
@Entity(name = "contracts")
@Table(name="contracts")
public class Contract
{
    /** Уникальный идентификатор контракта */
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    /** Описание условий контракта */
    @Column(name="description")
    private String description;

    /** Цена контракта */
    @Column(name="price")
    private double price;

    /** Клиент, подписавший контракт */
    @ManyToOne
    @JoinColumn (name = "client_id")
    private Client client;

    /** Менеджер, подписавший контракт */
    @ManyToOne
    @JoinColumn (name = "manager_id")
    private Manager manager;

    /** Рабочие, выполняющие условия контракта */
```

```

@ManyToMany(mappedBy = "contracts", fetch = FetchType.EAGER)
private List<Employee> workers;

/** Дата начала действия контракта */
@Column(name="begin")
private Date dateBegin;

/** Дата окончания действия контракта */
@Column(name="end")
private Date dateEnd;

/** Завершили ли выполнение договора */
@Column(name="is_end")
private boolean isEnd;

/** Стандартный конструктор контракта {@link Contract} */
public Contract() {}

/**
 * Конструктор - создание нового объекта {@link Contract}
 */
public Contract(String d, double p, Client c, Manager m, List<Employee> w, Date
b, Date e, boolean i)
{
    this.description = d;
    this.price = p;
    this.client = c;
    this.manager = m;
    this.workers = w;
    this.dateBegin = b;
    this.dateEnd = e;
    this.isEnd = i;
}

/**
 * Метод получения значения поля {@link Contract#id}
 * @return возвращает уникальный идентификатор контракта
 */
public int getId()
{
    return id;
}

/**

```

```

* Функция определения значения поля {@link Contract#id}
* @param newID - новый идентификатор
*/
public void setId(int newID)
{
    id = newID;
}

/**
* Метод определения значения поля {@link Contract#description}
* @param newDescription - описание контракта
*/
public void setDescription(String newDescription)
{
    description = newDescription;
}

/**
* Метод получения значения поля {@link Contract#description}
* @return возвращает описание контракта
*/
public String getDescription()
{
    return description;
}

/**
* Метод определения значения поля {@link Contract#price}
* @param newPrice - цена контракта
*/
public void setPrice(double newPrice)
{
    price = newPrice;
}

/**
* Метод получения значения поля {@link Contract#price}
* @return возвращает цену контракта
*/
public double getPrice()
{
    return price;
}

```

```

/**
 * Метод определения значения поля {@link Contract#client}
 * @param newClient - новый клиент контракта
 */
public void setClient(Client newClient)
{
    client = newClient;
}

/**
 * Метод получения значения поля {@link Contract#client}
 * @return возвращает клиента
 */
public Client getClient()
{
    return client;
}

/**
 * Метод определения значения поля {@link Contract#manager}
 * @param newManager - новый менеджер контракта
 */
public void setManager(Manager newManager)
{
    manager = newManager;
}

/**
 * Метод получения значения поля {@link Contract#manager}
 * @return возвращает менеджера
 */
public Manager getManager()
{
    return manager;
}

/**
 * Метод определения значения поля {@link Contract#dateBegin}
 * @param newDate - новая дата
 */
public void setDateBegin(Date newDate)
{
    dateBegin = newDate;
}

```

```

/**
 * Метод получения значения поля {@link Contract#dateBegin}
 * @return возвращает дату начала
 */
public Date getDateBegin()
{
    return dateBegin;
}

/**
 * Метод определения значения поля {@link Contract#dateEnd}
 * @param newDate - новая дата
 */
public void setDateEnd(Date newDate)
{
    dateEnd = newDate;
}

/**
 * Метод получения значения поля {@link Contract#dateEnd}
 * @return возвращает дату начала
 */
public Date getDateEnd()
{
    return dateEnd;
}

/**
 * Метод добавления рабочего, который выполняет контракт {@link
Contract#workers}
 * @param newWorker - новый рабочий
 */
public void addWorker(Employee newWorker)
{
    workers.add(newWorker);
}

/**
 * Метод удаления рабочего, который выполняет контракт {@link
Contract#workers}
 * @param worker - рабочий
 */
public void removeWorker(Employee worker)

```



```

    {
        workers.remove(worker);
    }

    /**
     * Метод установки рабочих, которые выполняют контракт {@link
Contract#workers}
     * @param w - список рабочих
     */
    public void setWorkers(List<Employee> w)
    {
        workers = w;
    }

    /**
     * Метод получения значения поля {@link Contract#workers}
     * @return возвращает всех рабочих, выполняющих контракт
     */
    public List<Employee> getWorkers()
    {
        return workers;
    }

    /**
     * Метод получения значения поля {@link Contract#isEnd}
     * @return возвращает состояние договора
     */
    public boolean getIsEnd()
    {
        return isEnd;
    }

    /**
     * Функция определения значения поля {@link Contract#id}
     * @param i - новое состояние
     */
    public void setIsEnd(boolean i)
    {
        isEnd = i;
    }

    /**
     * Функция получения всей информации об объекте
     * @return - массив строк с данными

```

```

*/
public String[] toTableFormat()
{
    DateFormat df = new SimpleDateFormat("yyyy-MM-dd");

    return new String[] {
        String.valueOf(id),
        description,
        String.valueOf(price),
        String.valueOf(client.getId()) + " " + client.getName() + " " +
client.getSurname(),
        String.valueOf(manager.getId()) + " " + manager.getName() + " " +
manager.getSurname(),
        df.format(dateBegin),
        df.format(dateEnd),
        isEnd ? "Выполнено" : "В процессе"
    };
}
}

```