Министерство науки и высшего образования Российской Федерации   
Федеральное государственное бюджетное образовательное   
учреждение высшего образования   
«Алтайский государственный технический   
университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Савченко В.В.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Отчёт

по лабораторной работе № 03

«Первая итерация проекта системы»

по дисциплине «Объектно-ориентированное программирование»

Студент группы ПИ-92 Гулин А.Н.

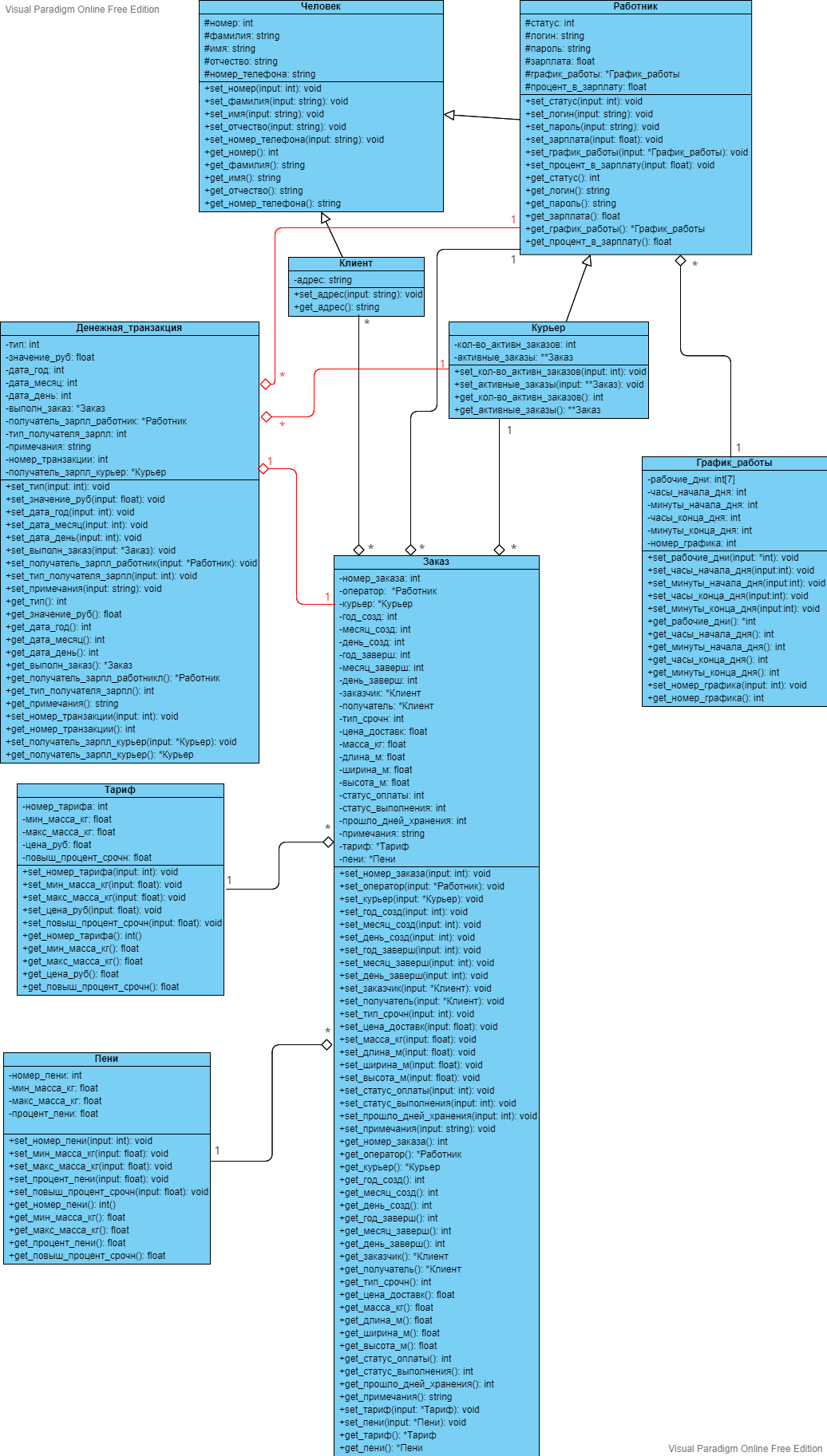
Преподаватель Савченко В.В.

Барнаул 2022

**Задание**

|  |  |  |  |
| --- | --- | --- | --- |
| № | ФИО студента  группы ПИ-92 | Прикладная область | Задание |
| 206 | Гулин Алексей Николаевич | Системы навигации и трассировки маршрутов | Проектирование автоматизированной системы курьерской службы в городских условиях плотной застройки |

**Схема классов**



**Описание классов**

1. **Класс** «Человек». Является основой для классов работников и клиентов, содержит наиболее базовую информацию о человеке.  
   **Поля**: Уникальный номер (целое число), фамилия (строка), имя (строка), отчество (строка), номер телефона (строка).  
   **Методы**: set и get всех соответствующих полей.
2. **Класс** «Работник». Наследник класса «Человек». Содержит основную информацию об операторе либо администраторе (либо о курьере после наследования в отдельный класс).  
   **Поля**: статус («работает», «отпуск», «больничный», «уволен» и т.п. – некоторое целое число), логин (строка), пароль (строка), текущее значение зарплаты в руб. (число с плавающей точкой), указатель на объект класса «График работы» (для этого работника), процент в счёт зарплаты с заказов (число с плавающей точкой).  
   **Методы**: set и get всех соответствующих полей.
3. **Класс** «График работы». Информация о графике работы отдельного работника.  
   **Поля**: Уникальный номер графика (целое число), рабочие дни недели   
   (В виде числового массива из 7 элементов. Если элемент равен 1, то день является рабочим, если равен 0, то является нерабочим), время начала рабочего дня (X часов Y минут), время окончания рабочего дня   
   (X часов Y минут).  
   **Методы**: set и get всех соответствующих полей.
4. **Класс** «Курьер». Наследник класса «Работник». Содержит основную информацию о курьере.  
   **Поля**: кол-во активных заказов (целое число), массив указателей на объекты класса «Заказ» (те заказы, что сейчас активны для данного курьера).  
   **Методы**: set и get всех соответствующих полей.
5. **Класс** «Денежная транзакция». Информация об отдельной транзакции.  
   **Поля**: Номер транзакции (целое число), тип («выполненный заказ», «зарплата», «некоторая выручка», «некоторые расходы» - некоторое целое число), значение в рублях (число с плавающей точкой), дата транзакции, указатель на объект «Заказ» (на выполненный заказ. Используется,   
   если тип = «выполненный заказ»), тип получателя зарплаты («администратор», «оператор», «курьер» - некоторое целое число, используется,   
   если тип = «зарплата»), указатель на получателя зарплаты – объект класса «Работник» (если тип получателя равен «оператор» или «администратор»,   
   и если тип транзакции = «зарплата»), указатель на получателя зарплаты – объект класса «Курьер» (если тип получателя равен «курьер», и если   
   тип транзакции = «зарплата»), примечания (строка).  
   **Методы**: set и get всех соответствующих полей.
6. **Класс** «Клиент». Наследник класса «Человек». Содержит информацию о клиенте заказа (заказчике, получателе).  
   **Поля**: адрес (строка).  
   **Методы**: set и get всех соответствующих полей.
7. **Класс** «Заказ». Содержит всю информацию о заказе по доставке посылки.  
   **Поля**: Номер заказа (целое число), указатель на объект класса «Работник» -   
   на оператора заказа (заполняется при создании), указатель на объект класса «Курьер» - на курьера заказа (заполняется, когда курьер выбрал заказ на исполнение), дата создания, дата завершения, указатель на объект класса «Клиент» - на заказчика, указатель на объект класса «Клиент» -   
   на получателя, тип срочности («обычный» или «срочно» – некоторое целое число), стоимость доставки в рублях (число с плавающей точкой), указатель   
   на объект класса «Тариф», указатель на объект класса «Пени», масса посылки в кг (число с плавающей точкой), длина посылки в м (число с плавающей точкой), ширина посылки в м (число с плавающей точкой), высота посылки   
   в м (число с плавающей точкой), статус оплаты («не оплачено», «оплачено наличными», «оплачено через терминал» – некоторое целое число), статус выполнения («ожидает выполнения», «курьер забирает посылку», «доставляется», «доставлено», «хранится», «отменено» – некоторое целое число), прошедшее время хранения в днях (целое число), примечания (строка).  
   **Методы**: set и get всех соответствующих полей.
8. **Класс** «Тариф». Содержит информацию о тарифе по доставке посылки определённой массы и срочности.  
   **Поля**: Номер тарифа (целое число), минимальная масса посылки в кг (число с плавающей точкой), максимальная масса посылки в кг (число с плавающей точкой), стоимость в рублях (число с плавающей точкой), процент повышения для срочного заказа (число с плавающей точкой).  
   **Методы**: set и get всех соответствующих полей.

1. **Класс** «Пени». Содержит информацию о пени хранения посылки определённой массы в день.  
   **Поля**: Номер пени (целое число), минимальная масса посылки в кг (число с плавающей точкой), максимальная масса посылки в кг (число с плавающей точкой), процент пени от стоимости заказа (число с плавающей точкой).  
   **Методы**: set и get всех соответствующих полей.

**Реализация классов в коде**

#define WEEK\_DAYS 7 //кол-во дней в неделе

//объявление классов

class Human; //Человек

class Worker; //Работник

class Schedule; //График работы

class Courier; //Курьер

class Financial\_transaction; //Денежная (финансовая) транзакция

class Client; //Клиент

class Order; //Заказ

class Rate; //Тариф по доставке посылки

class Penalty; //Пени за хранение посылки в день

class Human //Человек

{

protected:

int num; //уникальный номер в системе

string surname; //фамилия

string name; //имя

string patronymic; //отчество

string phone; //номер телефона

public:

Human() //конструктор

{}

~Human() //деструктор

{}

//методы set и get для полей класса

void set\_num(int input)

{

num = input;

}

void set\_surname(string input)

{

surname = input;

}

void set\_name(string input)

{

name = input;

}

void set\_patronymic(string input)

{

patronymic = input;

}

void set\_phone(string input)

{

phone = input;

}

////

int get\_num()

{

return num;

}

string get\_surname()

{

return surname;

}

string get\_name()

{

return name;

}

string get\_patronymic()

{

return patronymic;

}

string get\_phone()

{

return phone;

}

};

class Worker : public Human //Работник

{

protected:

int status; //статус («работает», «отпуск», «больничный», «уволен» и т.п.)

string login; //логин

string password; //пароль

float salary\_now; //текущее значение зарплаты в руб.

Schedule\* own\_schedule; //указатель на график работы

float percent\_to\_salary; //процент в счёт зарплаты с заказов

public:

Worker() //конструктор

{}

~Worker() //деструктор

{}

//методы set и get для полей класса

void set\_status(int input)

{

status = input;

}

void set\_login(string input)

{

login = input;

}

void set\_password(string input)

{

password = input;

}

void set\_salary\_now(float input)

{

salary\_now = input;

}

void set\_own\_schedule(Schedule\* input)

{

own\_schedule = input;

}

void set\_percent\_to\_salary(float input)

{

percent\_to\_salary = input;

}

////

int get\_status()

{

return status;

}

string get\_login()

{

return login;

}

string get\_password()

{

return password;

}

float get\_salary\_now()

{

return salary\_now;

}

Schedule\* get\_own\_schedule()

{

return own\_schedule;

}

float get\_percent\_to\_salary()

{

return percent\_to\_salary;

}

};

class Schedule //График работы

{

private:

int num\_schedule; //уникальный номер графика

int working\_days[WEEK\_DAYS]; //рабочие дни недели

int start\_hours; //время начала рабочего дня (часы)

int start\_minutes; //время начала рабочего дня (минуты)

int end\_hours; //время окончания рабочего дня (часы)

int end\_minutes; //время окончания рабочего дня (минуты)

public:

Schedule() //конструктор

{}

~Schedule() //деструктор

{}

//методы set и get для полей класса

void set\_num\_schedule(int input)

{

num\_schedule = input;

}

void set\_working\_days(int \*input)

{

for(int i=0; i<WEEK\_DAYS; i++)

{

working\_days[i] = input[i];

}

}

void set\_start\_hours(int input)

{

start\_hours = input;

}

void set\_start\_minutes(int input)

{

start\_minutes = input;

}

void set\_end\_hours(int input)

{

end\_hours = input;

}

void set\_end\_minutes(int input)

{

end\_minutes = input;

}

////

int get\_num\_schedule()

{

return num\_schedule;

}

int\* get\_working\_days()

{

return working\_days;

}

int get\_start\_hours()

{

return start\_hours;

}

int get\_start\_minutes()

{

return start\_minutes;

}

int get\_end\_hours()

{

return end\_hours;

}

int get\_end\_minutes()

{

return end\_minutes;

}

};

class Courier : public Worker //Курьер

{

private:

int qty\_active\_orders; //кол-во активных заказов

Order\*\* active\_orders; //массив указателей на активные заказы

public:

Courier() //конструктор

{}

~Courier() //деструктор

{}

//методы set и get для полей класса

void set\_qty\_active\_orders(int input)

{

qty\_active\_orders = input;

}

void set\_active\_orders(Order \*\*input)

{

active\_orders = input;

}

////

int get\_qty\_active\_orders()

{

return qty\_active\_orders;

}

Order\*\* get\_active\_orders()

{

return active\_orders;

}

};

class Financial\_transaction //Денежная (финансовая) транзакция

{

private:

int num\_transaction; //уникальный номер транзакции

int type\_transaction; //тип («выполненный заказ», «зарплата», «некоторая выручка», «некоторые расходы»)

float value; //значение в рублях

int date\_day; //дата транзакции (день)

int date\_month; //дата транзакции (месяц)

int date\_year; //дата транзакции (год)

Order\* complete\_order; //указатель на выполненный заказ (используется, если тип = «выполненный заказ»)

int type\_recipient; //тип получателя зарплаты («администратор», «оператор», «курьер»; используется, если тип = «зарплата»)

Worker\* recipient\_w; //указатель на получателя зарплаты - работника (используется, если тип = «зарплата» и тип получателя = «администратор» или «оператор»)

Courier\* recipient\_c; //указатель на получателя зарплаты - курьера (используется, если тип = «зарплата» и тип получателя = «курьер»)

string notes; //примечания транзакции

public:

Financial\_transaction() //конструктор

{}

~Financial\_transaction() //деструктор

{}

//методы set и get для полей класса

void set\_num\_transaction(int input)

{

num\_transaction = input;

}

void set\_type\_transaction(int input)

{

type\_transaction = input;

}

void set\_value(float input)

{

value = input;

}

void set\_date\_day(int input)

{

date\_day = input;

}

void set\_date\_month(int input)

{

date\_month = input;

}

void set\_date\_year(int input)

{

date\_year = input;

}

void set\_complete\_order(Order\* input)

{

complete\_order = input;

}

void set\_type\_recipient(int input)

{

type\_recipient = input;

}

void set\_recipient\_w(Worker\* input)

{

recipient\_w = input;

}

void set\_recipient\_c(Courier\* input)

{

recipient\_c = input;

}

void set\_notes(string input)

{

notes = input;

}

////

int get\_num\_transaction()

{

return num\_transaction;

}

int get\_type\_transaction()

{

return type\_transaction;

}

float get\_value()

{

return value;

}

int get\_date\_day()

{

return date\_day;

}

int get\_date\_month()

{

return date\_month;

}

int get\_date\_year()

{

return date\_year;

}

Order\* get\_complete\_order()

{

return complete\_order;

}

int get\_type\_recipient()

{

return type\_recipient;

}

Worker\* get\_recipient\_w()

{

return recipient\_w;

}

Courier\* get\_recipient\_c()

{

return recipient\_c;

}

string get\_notes()

{

return notes;

}

};

class Client : public Human //Клиент

{

private:

string address; //адрес

public:

Client() //конструктор

{}

~Client() //деструктор

{}

//методы set и get для полей класса

void set\_address(string input)

{

address = input;

}

////

string get\_address()

{

return address;

}

};

class Order //Заказ

{

private:

int num\_order; //уникальный номер заказа

Worker\* its\_operator; //указатель на оператора заказа

Courier\* its\_courier; //указатель на курьера заказа

int create\_day; //дата создания заказа (день)

int create\_month; //дата создания заказа (месяц)

int create\_year; //дата создания заказа (год)

int finish\_day; //дата завершения заказа (день)

int finish\_month; //дата завершения заказа (месяц)

int finish\_year; //дата завершения заказа (год)

Client\* customer; //указатель на заказчика

Client\* recipient; //указатель на получателя

Rate\* its\_rate; //указатель на тариф

Penalty\* its\_penalty; //указатель на пени

int type\_urgent; //тип срочности («обычный» или «срочно»)

float price; //стоимость доставки в рублях

float weight; //масса посылки в кг

float length; //длина посылки в м.

float width; //ширина посылки в м.

float height; //высота посылки в м.

int payment\_status; //статус оплаты («не оплачено», «оплачено наличными», «оплачено через терминал»)

int execution\_status; //статус выполнения («ожидает выполнения», «курьер забирает посылку», «доставляется», «доставлено», «хранится», «отменено»)

int storage\_time; //прошедшее время хранения в днях

string notes; //примечания заказа

public:

Order() //конструктор

{}

~Order() //деструктор

{}

//методы set и get для полей класса

void set\_num\_order(int input)

{

num\_order = input;

}

void set\_its\_operator(Worker\* input)

{

its\_operator = input;

}

void set\_its\_courier(Courier\* input)

{

its\_courier = input;

}

void set\_create\_day(int input)

{

create\_day = input;

}

void set\_create\_month(int input)

{

create\_month = input;

}

void set\_create\_year(int input)

{

create\_year = input;

}

void set\_finish\_day(int input)

{

finish\_day = input;

}

void set\_finish\_month(int input)

{

finish\_month = input;

}

void set\_finish\_year(int input)

{

finish\_year = input;

}

void set\_customer(Client\* input)

{

customer = input;

}

void set\_recipient(Client\* input)

{

recipient = input;

}

void set\_its\_rate(Rate\* input)

{

its\_rate = input;

}

void set\_its\_penalty(Penalty\* input)

{

its\_penalty = input;

}

void set\_type\_urgent(int input)

{

type\_urgent = input;

}

void set\_price(float input)

{

price = input;

}

void set\_weight(float input)

{

weight = input;

}

void set\_length(float input)

{

length = input;

}

void set\_width(float input)

{

width = input;

}

void set\_height(float input)

{

height = input;

}

void set\_payment\_status(int input)

{

payment\_status = input;

}

void set\_execution\_status(int input)

{

execution\_status = input;

}

void set\_notes(string input)

{

notes = input;

}

////

int get\_num\_order()

{

return num\_order;

}

Worker\* get\_its\_operator()

{

return its\_operator;

}

Courier\* get\_its\_courier()

{

return its\_courier;

}

int get\_create\_day()

{

return create\_day;

}

int get\_create\_month()

{

return create\_month;

}

int get\_create\_year()

{

return create\_year;

}

int get\_finish\_day()

{

return finish\_day;

}

int get\_finish\_month()

{

return finish\_month;

}

int get\_finish\_year()

{

return finish\_year;

}

Client\* get\_customer()

{

return customer;

}

Client\* get\_recipient()

{

return recipient;

}

Rate\* get\_its\_rate()

{

return its\_rate;

}

Penalty\* get\_its\_penalty()

{

return its\_penalty;

}

int get\_type\_urgent()

{

return type\_urgent;

}

float get\_price()

{

return price;

}

float get\_weight()

{

return weight;

}

float get\_length()

{

return length;

}

float get\_width()

{

return width;

}

float get\_height()

{

return height;

}

int get\_payment\_status()

{

return payment\_status;

}

int get\_execution\_status()

{

return execution\_status;

}

string get\_notes()

{

return notes;

}

};

class Rate //Тариф по доставке посылки

{

private:

int num\_rate; //уникальный номер тарифа

float min\_weight; //минимальная масса посылки в кг

float max\_weight; //максимальная масса посылки в кг

float cost; //стоимость в рублях

float percent\_increase; //процент повышения стоимости от обычной для срочного заказа

public:

Rate() //конструктор

{}

~Rate() //деструктор

{}

//методы set и get для полей класса

void set\_num\_rate(int input)

{

num\_rate = input;

}

void set\_min\_weight(float input)

{

min\_weight = input;

}

void set\_max\_weight(float input)

{

max\_weight = input;

}

void set\_cost(float input)

{

cost = input;

}

void set\_percent\_increase(float input)

{

percent\_increase = input;

}

////

int get\_num\_rate()

{

return num\_rate;

}

float get\_min\_weight()

{

return min\_weight;

}

float get\_max\_weight()

{

return max\_weight;

}

float get\_cost()

{

return cost;

}

float get\_percent\_increase()

{

return percent\_increase;

}

};

class Penalty //Пени за хранение посылки в день

{

private:

int num\_penalty; //уникальный номер пени

float min\_weight; //минимальная масса посылки в кг

float max\_weight; //максимальная масса посылки в кг

float percent\_penalty; //процент пени от стоимости заказа

public:

Penalty() //конструктор

{}

~Penalty() //деструктор

{}

//методы set и get для полей класса

void set\_num\_penalty(int input)

{

num\_penalty = input;

}

void set\_min\_weight(float input)

{

min\_weight = input;

}

void set\_max\_weight(float input)

{

max\_weight = input;

}

void set\_percent\_penalty(float input)

{

percent\_penalty = input;

}

////

int get\_num\_penalty()

{

return num\_penalty;

}

float get\_min\_weight()

{

return min\_weight;

}

float get\_max\_weight()

{

return max\_weight;

}

float get\_percent\_penalty()

{

return percent\_penalty;

}

};

**Возможные расширения системы**

* Отслеживание координат местоположения курьеров (доп. поля в классе «Курьер»).
* Координаты пункта отправки и пункта назначения (доп. поля в классе «Заказ»).
* Класс «Маршрут». Совокупность точек на карте (их координат) от местоположения курьера до активных заказов.
* Учитывать время, когда получатель может забрать заказ (с A ч. X мин.   
  до B ч. Y мин.) (доп. поля в классе «Заказ»).
* Класс «Чек-лист» для заказа. Содержит несколько строк, последовательно описывающих действия курьера (например, заполнить бланк, сфотографировать клиента и др.) для заказа, каждая строка может быть отмечена как выполненная или невыполненная. Заказ может быть отмечен как завершённый, когда все строки в чек-листе отмечены как выполненные.