Работа со строками в Java.

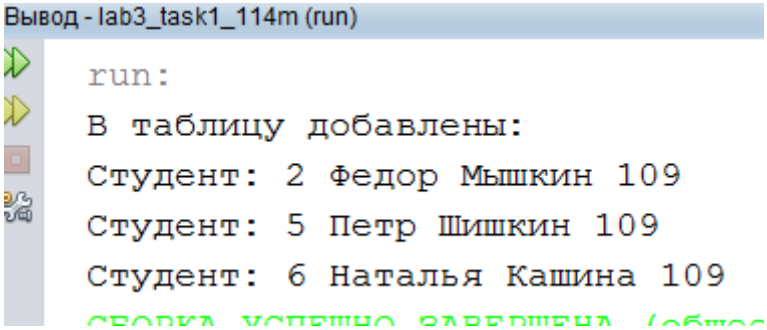
Задача 1. Создайте программу, формирующую SQL-инструкцию для вставки строк в таблицу TGroupSelected <idStudent, firstName, lastName, idGroup> из таблицы TStudent <idStudent, firstName, lastName, idGroup, dolgCount > для студентов, относящихся к определенной группе (строковый параметр) и имеющих количество долгов (целочисленный параметр), превышающее заданное значение. В SQL-инструкции строковые значения должны быть заключены в одинарные кавычки.

1. **Код программы:**

```
public class Lab3_task1_114m {

    //JDBC драйвер для работы с postgresql
    static String driver = "org.postgresql.Driver";
    //данные для соединения с БД
    static String connection =
"jdbc:postgresql://localhost:5432/basejava";
    static String user = "postgres";
    static String password = "qazxsw";

    public static void main(String[] args) {
        try {
            //вызов метода добавления данных из одной табл. в
другую с условием
            insertWhereTable();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    //метод соединение с БД
    private static Connection getDBConnection() {
        Connection dbConnection = null;
        try {
            Class.forName(driver); //указываем JDBC драйвер
        } catch (ClassNotFoundException e) {
            System.out.println(e.getMessage());
        }
        try {
            //JDBC драйвер обеспечивает соединение с базой
данных
            dbConnection =
DriverManager.getConnection(connection, user, password);
            return dbConnection;
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
        return dbConnection;
    }
```

```
    private static void insertWhereTable() throws SQLException {
        Connection dbConnection = null; //соединение
        Statement statement = null; //создание запросов
        String id_Group = "109"; //номер группы
        int dolgCount = 1; //кол-во долгов
        //sql запрос получения всех данных из табл.
        String selectTableSQL = "SELECT * FROM
                                        \"T_GroupSelected\" ";
        //sql запрос добавления данных из одной таблицы в другую
        //с условием, что номер группы = id_Group и кол-во
        долгов = dolgCount, заданным выше в коде
        String insertTableSQL = "INSERT INTO \"T_GroupSelected\"
        (\"id_Student\", \"firstName\", \"lastName\",
        \"id_Group\")\n"
                + "   SELECT \"id_Student\", \"firstName\",
                \"lastName\", \"id_Group\"\n"
                + "   FROM \"T_Student\"\n"
                + "    WHERE \"id_Group\"='" + id_Group + "' AND
                \"dolgCount\">" + dolgCount;
        try {
            dbConnection = getDBConnection(); //метод соединения
        с БД
            //создание объекта для отправки инструкций SQL в
базу данных
            statement = dbConnection.createStatement();
            //метод для выполнения команд SQL добавления
            statement.executeUpdate(insertTableSQL);
            //результаты запроса выполнения команды SELECT
            ResultSet rs =
                   statement.executeQuery(selectTableSQL);
            System.out.println("В таблицу добавлены:");
            //выборка из результатов
            while (rs.next()) {
                int id = rs.getInt(1); //id_Student
                String rsFirstName = rs.getString(2);
               //firstName
                String rsLastName = rs.getString(3); //lastName
                String rsId_Group = rs.getString(4); //id_Group
                //вывод в консоль добавленных из табл. в табл.
               студентов
                System.out.println("Студент: " + id + " " +
               rsFirstName + " " + rsLastName + " " +
               rsId_Group);
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
        dbConnection.close();
    }
```

Вывод:



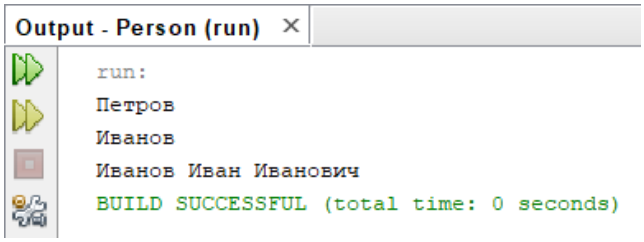
2. Создайте класс Person с функцией, возвращающей Фамилию и инициалы. Функция должна учитывать возможность отсутствия значений в полях Имя и Отчество. Оптимизируйте программу с точки зрения производительности.

Код программы:

```
public class Person {
    //Создание переменных класса
    private String name, surname, patronymic;
    //конструктор для фамилии, если имя и отчество отсутвуют
    public Person(String surname) {
        this.surname = surname;
    }
    //конструктор ФИО
    public Person(String name, String surname, String
patronymic) {
        this.name = name;
        this.surname = surname;
        this.patronymic = patronymic;
    }
    //Метод для учета возможности отсутствия значений в полях
Имя и Отчество
    public String getFio(){
        //создаем объект класса StringBuilder для реализации
изменяемых строк
        StringBuilder sb = new StringBuilder(surname);
        if(name != null && ! name.equals(""))
            sb.append(" ").append(name);
        if(patronymic != null && ! patronymic.equals(""))
            sb.append(" ").append(patronymic);
        return sb.toString();
    }

    public static void main(String[] args) {
        Person p1 = new Person("Петров");//Только фамилия
        Person p2 = new Person("", "Иванов", ""); //Отсутствует
Имя и Отчество
        Person p3 = new Person("Иван", "Иванов", "Иванович");
        //Вывод данных на экран
        System.out.println(p1.getFio());
        System.out.println(p2.getFio());
```

```
        System.out.println(p3.getFio());
    }
}
```

Вывод:



3. Улучшите класс Address, который из строки формата "Странad Регионd Городd Улицаd Домd Корпусd Квартира" (d – разделитель, например, «запятая») извлекает соответствующие части и записывает их в соответствующие поля. Реализуйте две версии этого метода:

- Разделитель – только запятая (используйте метод split());
- Разделитель – любой из символов ,.;- (с помощью класса StringTokenizer).

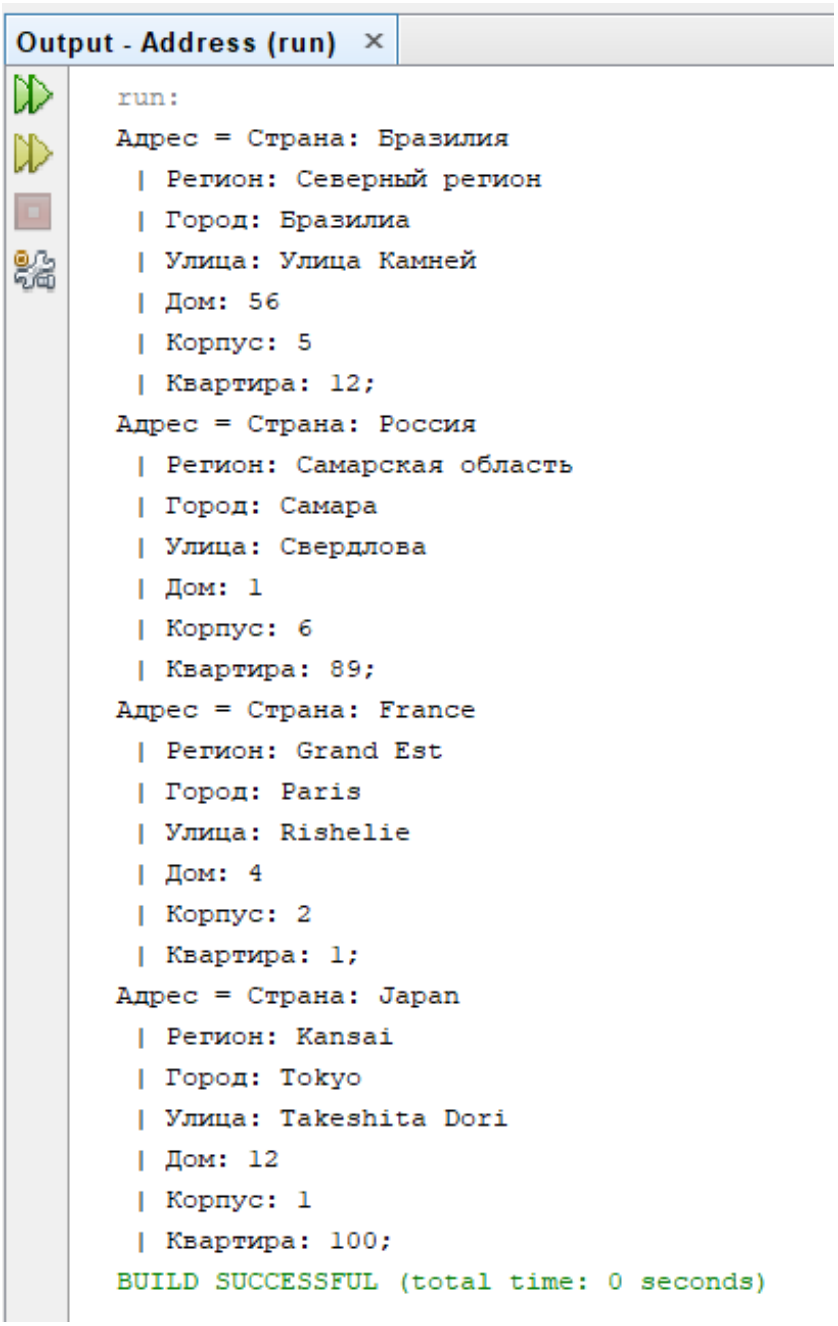
Имейте в виду, что в начале и конце разобранной части адреса не должно быть пробелов. Все поля адреса должны быть строками. Создайте тестовый класс с не менее чем четырьмя адресными строками.

Код программы:

```
public class Address {

    private String country, region, city, street, house,
building, flat;

    public static void main(String[] args) {
        //Первый и четвертый адреса тестируют наличие
разделителей ",.;"
        Address first = new Address("Бразилия,    Северный
регион; Бразилиа.. Улица Камней  , 56  ; 5 .; 12", true);
        //Второй и третий адреса тестируют строку при наличии
только запятой ,
        Address second = new Address("  Россия ,    Самарская
область, Самара  , Свердлова  , 1  , 6    , 89", false);
        Address third = new Address(" France  , Grand Est
, Paris, Rishelie  , 4  , 2, 1", false);
        Address fourth = new Address("   Japan  ;   Kansai ;
Tokyo . Takeshita Dori  , 12  ; 1; 100", true);

        System.out.println(first);
        System.out.println(second);
        System.out.println(third);
        System.out.println(fourth);
    }
```

```
    public Address(String stringOfAddress, boolean divider){
        if(stringOfAddress == null) throw new
NullPointerException();
        String[] s;
        if(divider){
            //разделитель - любой из символов ,.;- (класс
StringTokenizer)
            StringTokenizer st = new
StringTokenizer(stringOfAddress, ",.;");
            //Текстовый массив строк, используемый для
разложения на слова
            s = new String[st.countTokens()];//подсчет
элементов, которые осталось разобрать и возвратить в качестве
результата
            int i = 0;
            //После окончания элементов в текущей строке
переходит к следующей
            while(st.hasMoreTokens()) s[i++] = st.nextToken();
        }else{
            //разделитель - только запятая (используемый метод
split())
            s = stringOfAddress.split(",");
        }
        if(s.length < 7) throw new
IllegalArgumentException("Введен неполный адрес!");
        //trim() удаляет все начальные и конечные символы
пробела из текущей строки
        country = s[0].trim();
        region = s[1].trim();
        city = s[2].trim();
        street = s[3].trim();
        house = s[4].trim();
        building = s[5].trim();
        flat = s[6].trim();
    }

    @Override
    public String toString() {
        return "Адрес = " + "Страна: " + country + "\n" + " |
Регион: " + region + "\n" +
                " | Город: " + city + "\n" + " | Улица: " +
street + "\n" + " | Дом: " +
                house + "\n" + " | Корпус: " + building + "\n" +
" | Квартира: " + flat + ';';
    }
}
```

Вывод:



4. Реализуйте класс Shirt:

Метод toString() выводит описание и значения полей, расположенные построчно.

Также дан строковый массив:

```
shirts[0] = "S001,Black Polo Shirt,Black,XL";
shirts[1] = "S002,Black Polo Shirt,Black,L";
shirts[2] = "S003,Blue Polo Shirt,Blue,XL";
shirts[3] = "S004,Blue Polo Shirt,Blue,M";
shirts[4] = "S005,Tan Polo Shirt,Tan,XL";
shirts[5] = "S006,Black T-Shirt,Black,XL";
shirts[6] = "S007,White T-Shirt,White,XL";
shirts[7] = "S008,White T-Shirt,White,L";
```

```
shirts[8] = "S009,Green T-Shirt,Green,S";
shirts[9] = "S010,Orange T-Shirt,Orange,S";
shirts[10] = "S011,Maroon Polo Shirt,Maroon,S";
```

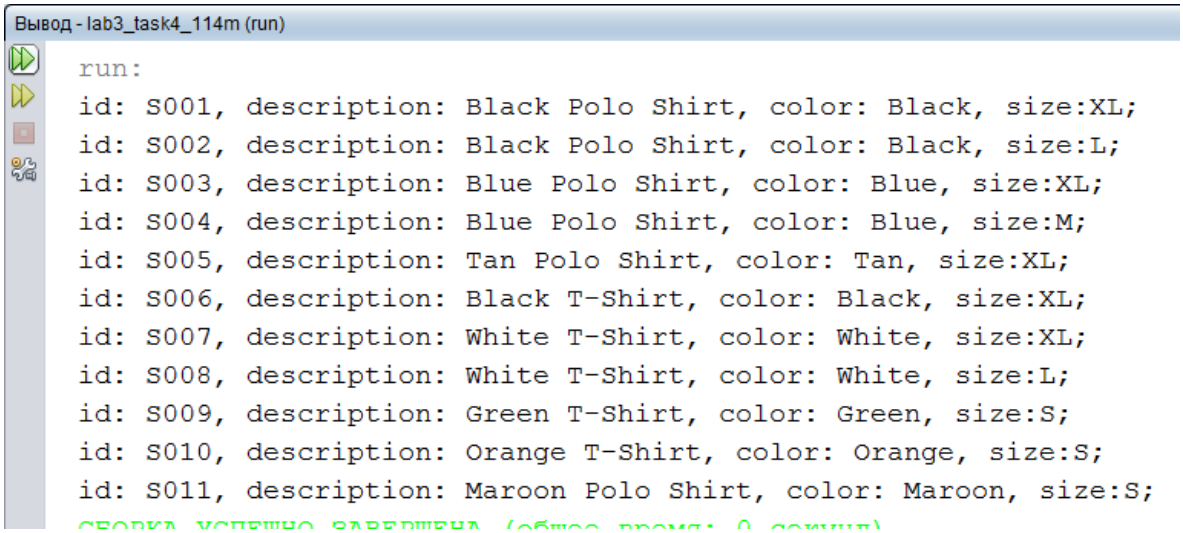
Преобразуйте строковый массив в массив класса Shirt и выведите его на консоль.

Код программы:

```
public class Lab3_task4_114m {
    public static void main(String[] args) {
        //создание строкового массива
        String[] shirts = new String[11];
        shirts[0] = "S001,Black Polo Shirt,Black,XL";
        shirts[1] = "S002,Black Polo Shirt,Black,L";
        shirts[2] = "S003,Blue Polo Shirt,Blue,XL";
        shirts[3] = "S004,Blue Polo Shirt,Blue,M";
        shirts[4] = "S005,Tan Polo Shirt,Tan,XL";
        shirts[5] = "S006,Black T-Shirt,Black,XL";
        shirts[6] = "S007,White T-Shirt,White,XL";
        shirts[7] = "S008,White T-Shirt,White,L";
        shirts[8] = "S009,Green T-Shirt,Green,S";
        shirts[9] = "S010,Orange T-Shirt,Orange,S";
        shirts[10] = "S011,Maroon Polo Shirt,Maroon,S";
        //создание массива класса Shirt
        Shirt[] newShirts = new Shirt[11];
        //перобразование строковый массив в массив класса Shirt
        //и вывыод его в консоль
        for (int i = 0; i < shirts.length; i++) {
            newShirts[i] = new Shirt(shirts[i]);
            //вывод строки Shirt переопределеным методом
toString
            System.out.println(newShirts[i].toString());
        }
    }

    static public class Shirt {
        private String id, description, color, size;

        public Shirt(String shirt) {
            //разбиение строки на подстроки
            String[] str = shirt.split(",");
            id = str[0];
            description = str[1];
            color = str[2];
            size = str[3];
        }
        //переопределение метода toString
        @Override
        public String toString() {
            return "id: " + id + ", description: " + description
+ ", color: " + color + ", size:" + size + ";";
        }
```

```
    }
}
```

Вывод:



5. Создайте класс, который принимает телефонный номер в одном из двух форматов:

+<Код страны><10-значный номер>, например, “+79175655655” или “+104289652211”,

либо

8<10-значный номер> для России, например, “89175655655”,

и преобразует его в формат:

+<Код страны><Три цифры>–<Три цифры>–<Четыре цифры>.

Код программы:

```
public class Lab3_task5_114m {

    public static void main(String[] args) {
        //вывод номеров, начинающихся с +
        System.out.println("before: +79175655655 after: "+new
Phone("+79175655655").toString());
        System.out.println("before: +104289652211 after: "+new
Phone("+104289652211").toString());
        //вывод номеров, начинающихся с 8
        System.out.println("before: 89175655655 after: "+new
Phone("89175655655").toString());
    }

    public static class Phone {
        String countryCode, firstThreenumb, threenumb, fournumb;

        public Phone(String phoneNumber) {
            //если номер начинается на +
            if (phoneNumber.charAt(0) == '+') {
```

```
            //код страны первые цифры без учета 10ти цифр с
конца
                countryCode = phoneNumber.substring(0,
phoneNumber.length() - 10);
            //если номер не начинается с +
            } else {
                countryCode = "+7"; //то код россии
            }
            //первые три цифры после кода страны
            firstThreenumb =
phoneNumber.substring(phoneNumber.length() - 10,
phoneNumber.length() - 7);
            threenumb =
phoneNumber.substring(phoneNumber.length() - 7,
phoneNumber.length() - 4); //далее четыре цифры
            fournumb =
phoneNumber.substring(phoneNumber.length() - 4); //последние три
цифры
        }
        //переопределение метода преобразования строки
        @Override
        public String toString() {
            return countryCode+firstThreenumb+"-"+threenumb+"-
"+fournumb;
        }
    }
}
```

Вывод: