**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 1**

по дисциплине:

«Визуальные средства разработки программных приложений»

на тему:

«Создание собственных классов и интерфейсов в *Java*»

Выполнил: студент гр. ИТП-31

Коркуц С. И.

Принял: ассистент

Гуменников Е.Д.

Гомель 2020

**Цель работы:** изучить способы создания собственных классов и интерфейсов в *Java.*

**Задание:** Вариант 7. Условия задания на рисунках 1 – 2.

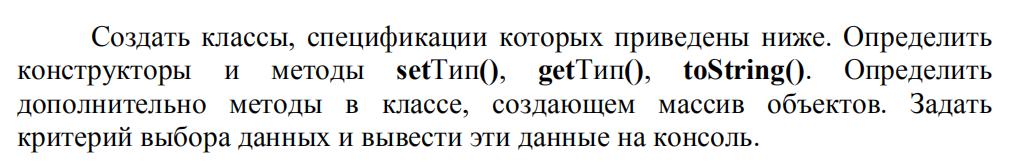
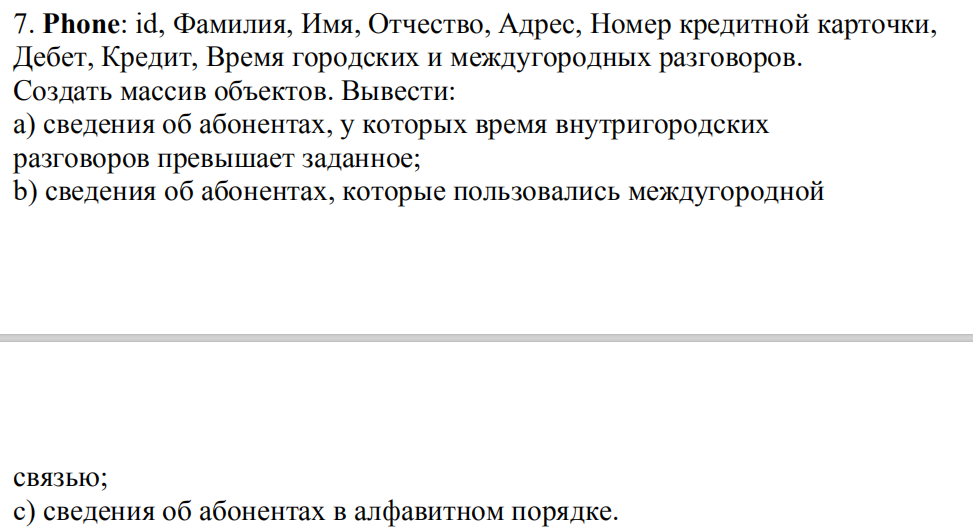


Рисунок 1 – Условие задания



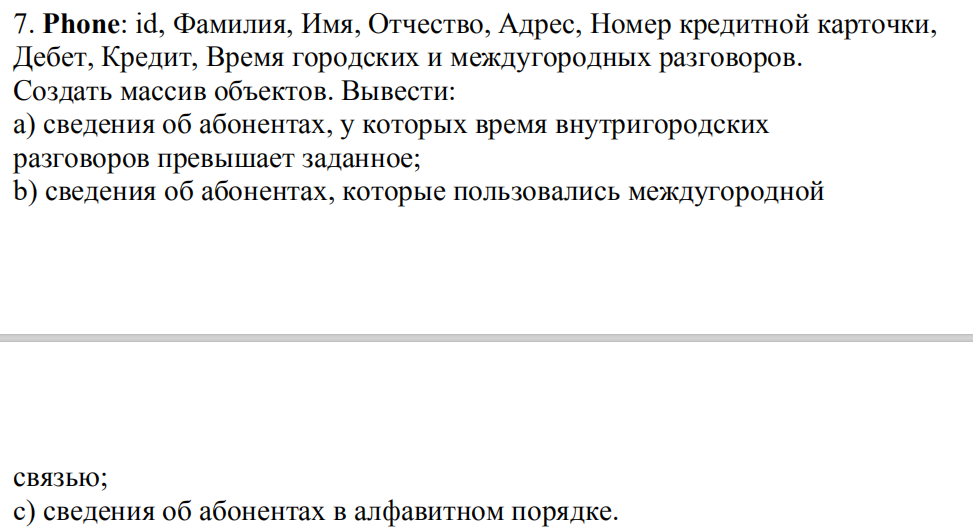


Рисунок 2 – Вариант задания

Структура созданных классов на рисунке 3.

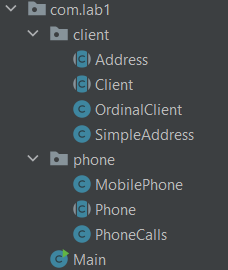


Рисунок 3 – Структура созданных классов

Изначально были созданы абстрактные классы для сущностей адрес, клиент и телефон. Далее были созданы конкретные реализации этих классов.

В классе Main было написано заполнение базы тестовыми значениями, далее были написаны методы, решающие данные задания.

Результат выполнения задания представлен на рисунке 4.

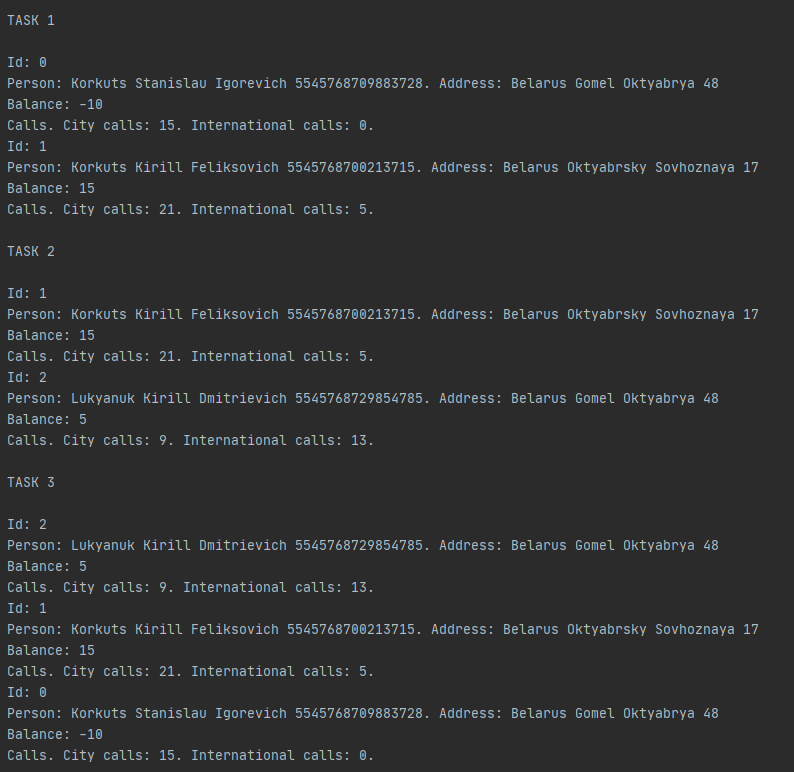


Рисунок 4 – Результат выполнения задания

**Вывод:** Были написаны абстрактные классы и их реализации соответственно поставленному заданию. Были написаны аксессоры и переопределения наследуемых методов.

**ПРИЛОЖЕНИЕ А**

Листинг созданных классов

package com.lab1.client;  
  
public abstract class Address {  
 private String country;  
 private String city;  
 private String street;  
 private int houseNumber;  
  
 public Address(String country, String city, String street, int houseNumber) {  
 this.country = country;  
 this.city = city;  
 this.street = street;  
 this.houseNumber = houseNumber;  
 }  
  
 public String getCountry() {  
 return country;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public String getStreet() {  
 return street;  
 }  
  
 public int getHouseNumber() {  
 return houseNumber;  
 }  
  
 @Override  
 public String toString() {  
 return String.format("Address: %s %s %s %d", country, city, street, houseNumber);  
 }  
}

package com.lab1.client;  
  
public abstract class Client {  
 private String firstName;  
 private String secondName;  
 private String lastName;  
 private String creditCardNumber;  
 private Address address;  
  
 public Client(String firstName, String secondName, String lastName, String creditCardNumber, Address address) {  
 this.firstName = firstName;  
 this.secondName = secondName;  
 this.lastName = lastName;  
 this.creditCardNumber = creditCardNumber;  
 this.address = address;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public String getSecondName() {  
 return secondName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public String getCreditCardNumber() {  
 return creditCardNumber;  
 }  
  
 public Address getAddress() {  
 return address;  
 }  
  
 @Override  
 public String toString() {  
 return String.format("Person: %s %s %s %s. %s", secondName, firstName, lastName, creditCardNumber, address);  
 }  
}

package com.lab1.client;  
  
public class OrdinalClient extends Client {  
 public OrdinalClient(String firstName, String secondName, String lastName, String creditCardNumber, Address address) {  
 super(firstName, secondName, lastName, creditCardNumber, address);  
 }  
}

package com.lab1.client;  
  
public class SimpleAddress extends Address {  
 public SimpleAddress(String country, String city, String street, int houseNumber) {  
 super(country, city, street, houseNumber);  
 }  
}

package com.lab1.phone;  
  
import com.lab1.client.Client;  
  
public abstract class Phone implements Comparable<Phone> {  
 private static int lastId = 0;  
 private int id;  
 private Client client;  
 private int accountBalance;  
 private PhoneCalls calls;  
  
 public Phone(Client client, int accountBalance, PhoneCalls calls) {  
 id = lastId++;  
 this.client = client;  
 this.accountBalance = accountBalance;  
 this.calls = calls;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public Client getPerson() {  
 return client;  
 }  
  
 public int getAccountBalance() {  
 return accountBalance;  
 }  
  
 public PhoneCalls getCalls() {  
 return calls;  
 }  
  
 @Override  
 public String toString() {  
 return String.format("Id: %d\n%s\nBalance: %d\n%s", id, client, accountBalance, calls);  
 }  
  
 @Override  
 public int compareTo(Phone o) {  
 return client.getLastName().compareTo(o.getPerson().getLastName());  
 }  
}

package com.lab1.phone;  
  
import com.lab1.client.Client;  
  
public class MobilePhone extends Phone {  
 public MobilePhone(Client client, int accountBalance, PhoneCalls calls) {  
 super(client, accountBalance, calls);  
 }  
}

package com.lab1.phone;  
  
public class PhoneCalls {  
 private int cityCalls;  
 private int internationalCalls;  
  
 public PhoneCalls(int cityCalls, int internationalCalls) {  
 this.cityCalls = cityCalls;  
 this.internationalCalls = internationalCalls;  
 }  
  
 public int getCityCalls() {  
 return cityCalls;  
 }  
  
 public int getInternationalCalls() {  
 return internationalCalls;  
 }  
  
 @Override  
 public String toString() {  
 return String.format("Calls. City calls: %d. International calls: %d.", cityCalls, internationalCalls);  
 }  
}

package com.lab1;  
  
import com.lab1.client.OrdinalClient;  
import com.lab1.client.SimpleAddress;  
import com.lab1.phone.MobilePhone;  
import com.lab1.phone.Phone;  
import com.lab1.phone.PhoneCalls;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Main {  
  
 public static void main(String[] args) {  
 List<Phone> phones = getPhones();  
 task1(phones, 14);  
 task2(phones);  
 task3(phones);  
 }  
  
 private static List<Phone> getPhones() {  
 List<Phone> phones = new ArrayList<>();  
 phones.add(new MobilePhone(new OrdinalClient("Stanislau", "Korkuts", "Igorevich",  
 "5545768709883728", new SimpleAddress("Belarus", "Gomel", "Oktyabrya",  
 48)), -10, new PhoneCalls(15, 0)));  
 phones.add(new MobilePhone(new OrdinalClient("Kirill", "Korkuts", "Feliksovich",  
 "5545768700213715", new SimpleAddress("Belarus", "Oktyabrsky", "Sovhoznaya",  
 17)), 15, new PhoneCalls(21, 5)));  
 phones.add(new MobilePhone(new OrdinalClient("Kirill", "Lukyanuk", "Dmitrievich",  
 "5545768729854785", new SimpleAddress("Belarus", "Gomel", "Oktyabrya",  
 48)), 5, new PhoneCalls(9, 13)));  
 return phones;  
 }  
  
 private static void task1(List<Phone> phones, int cityCalls) {  
 System.out.println("\nTASK 1\n");  
 for (Phone phone :  
 phones) {  
 if(phone.getCalls().getCityCalls() > cityCalls) {  
 System.out.println(phone);  
 }  
 }  
 }  
  
 private static void task2(List<Phone> phones) {  
 System.out.println("\nTASK 2\n");  
 for (Phone phone : phones) {  
 if(phone.getCalls().getInternationalCalls() > 0) {  
 System.out.println(phone);  
 }  
 }  
 }  
  
 private static void task3(List<Phone> phones) {  
 System.out.println("\nTASK 3\n");  
 phones.sort(null);  
 for (Phone phone : phones) {  
 System.out.println(phone);  
 }  
 }  
}