Министерство образования Калининградской области

государственное бюджетное учреждение Калининградской области

профессиональная образовательная организация

«Колледж информационных технологий и строительства»

(ГБУ КО ПОО «КИТиС»)

**Отчет по учебной практике**

УП.01 Разработка модулей программного обеспечения для компьютерных систем

по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Специальность 09.02.07 «Информационные системы и программирование»

Сроки прохождения практики:

с «12» октября 2022 г. по «01» ноября 2022 г.

Место практики  ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 4 курса,  группы ИСп 19-2к  Мурашкин Кирилл Владимирович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_              (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_              (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_        (подпись, дата) |

Калининград, 2022

Содержание

Введение………………………………………………………………………….3

Основная часть…………………………………………………………………...4

1. Задача 1.1……………………………………………………………………...4
2. Задача 1.2……………………………………………………………………...5
3. Задача 1.3……………………………………………………………………...9
4. Задача 1.4……………………………………………………………………..13
5. Задача 1.5……………………………………………………………………..15

6. Задача 1.6……………………………………………………………………..17

7. Разработка ER-диаграммы 2…………………………………………………18

Введение

Отчет составлен по учебной практике УП.01 Разработка модулей программного обеспечения для компьютерных систем.

   Цели:

1. Написать консольные задачи в среде разработки InteliJ IDEA.
2. Разработка АИС.
3. Разработка ER-диаграммы.
4. Сформировать «Техническое задание».
5. Проектирование и реализация интерфейса.
6. Разработать встроенную систему справочного руководства и программного документа «Руководство пользователя».
7. Проведение тестирования.
8. Обеспечить механизм авторизации и работы как минимум двух пользователей, разграничить права.

* 1. **Задача 1**

Найти корни квадратного уравнения. Параметры уравнения передавать с командной строкой

Main.java

package com.company;  
  
import java.util.Scanner;  
  
public class Task1 {  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 double a, b, c, D;  
 System.*out*.println("Введите переменные a, b и c квадратного уравнения ax+bx+c");  
 try {  
 a = sc.nextDouble();  
 b = sc.nextDouble();  
 c = sc.nextDouble();  
 D = b \* b - 4 \* a \* c;  
 if (D > 0) {  
 double x1, x2;  
 x1 = (-b - Math.*sqrt*(D)) / (2 \* a);  
 x2 = (-b + Math.*sqrt*(D)) / (2 \* a);  
 System.*out*.println("Корни уравнения: x1 = " + x1 + ", x2 = " + x2);  
 } else if (D == 0) {  
 double x;  
 x = -b / (2 \* a);  
 System.*out*.println("Уравнение имеет 1 корень: x = " + x);  
 } else {  
 System.*out*.println("Уравнение не имеет корней!");  
 }  
 } catch (NumberFormatException ex) {  
 System.*out*.println("Введите число!");  
 }  
 }  
}

Результат:

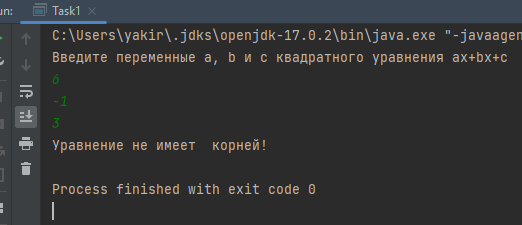


Рисунок – 1.

* 1. **Задача 2**

Patient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести: a) список пациентов, имеющих данный диагноз; b) список пациентов, номер медицинской карты которых находится в заданном интервале.

package com.company;  
  
public class Client {  
  
 private String clientSecondName;  
 private String clientFirstName;  
 private String clientPatronymic;  
 private String clientAddress;  
 private long clientCardNumber;  
 private double clientDebit;  
 private double clientCredit;  
 private String diagnoz;  
  
 public String getClientSecondName() {  
 return clientSecondName;  
 }  
  
 public String getClientDiagnoz() {  
 return diagnoz;  
 }  
  
 public Client(String clientSecondName, String clientFirstName, String clientPatronymic,  
 String clientAddress, long clientCardNumber, double clientDebit,  
 double clientCredit, String diagnoz) {  
  
 this.clientSecondName = clientSecondName;  
 this.clientFirstName = clientFirstName;  
 this.clientPatronymic = clientPatronymic;  
 this.clientAddress = clientAddress;  
 this.clientCardNumber = clientCardNumber;  
 this.clientDebit = clientDebit;  
 this.clientCredit = clientCredit;  
 this.diagnoz = diagnoz;  
 }  
  
 public String toString() {  
 return String.*format*("Client: %s %s %s", clientSecondName, clientFirstName, clientPatronymic) + "\n" + "Адрес проживания" + clientAddress +  
 String.*format*("\t\tCardNumber: %d\t Debit: %10.2f\t Credit: %10.2f.", clientCardNumber, clientDebit,  
 clientCredit) + "\n" + "Заболевание: "  
 + diagnoz;  
 }  
}

package com.company;  
  
  
import java.util.ArrayList;  
  
  
public class ClientInformation {  
 public static void main(String[] args) {  
  
 ClientNumber operator = new ClientNumber();  
 operator.addClient(new Client("Зябликов", "Артур", "Петрович",  
 "Минск, Кальварийская 21, кв.17.", 8365479658741236L, 8965.63,  
 300.2, "Сальмонелиоз"));  
 operator.addClient(new Client("Литехин", "Максим", "Андреевич",  
 "Минск, Пушкина 83, кв.45.", 1458796325412358L, 276.50,  
 856.35, "Бронхит"));  
 operator.addClient(new Client("Андропов", "Петр", "Максимович",  
 "Минск, Рафиева 16, кв.89.", 2354789632145698L, 5659.56,  
 46.34, "Бронхит"));  
 operator.addClient(new Client("Копытина", "Анастасия", "Сегеевна",  
 "Минск, Машиностроителей 10, кв.23.", 8789654789654214L, 294.35,  
 850.35, "Сердечная недостаточность"));  
 operator.addClient(new Client("Мормышкина", "Зинаида", "Алексеевна",  
 "Минск, Победителей 156, кв.36.", 8963214769854126L, 336.98,  
 45.75, "Бронхит"));  
 System.*out*.println("Клиенты по именам:");  
  
 ArrayList<Client> listName = operator.getClientListSortByABC();  
 for (Client client : listName) {  
 System.*out*.println(client);  
 }  
 System.*out*.println("\n\nКлиенты с заболеванием Бронхит");  
 ArrayList<Client> listInternational = operator.getClientListSortWithDiagnoz("Бронхит");  
 for (Client client : listInternational) {  
 System.*out*.println(client);  
 }  
 }  
}

package com.company;  
  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;  
  
public class ClientNumber {  
 private ArrayList<Client> clientList = new ArrayList<>();  
  
 void addClient(Client client) {  
 clientList.add(client);  
 }  
  
 public ArrayList<Client> getClientListSortWithDiagnoz(String diagnoz) {  
 ArrayList<Client> clientArrayList = new ArrayList<>();  
 for (Client client : clientList) {  
 if (client.getClientDiagnoz() == diagnoz) {  
 clientArrayList.add(client);  
 }  
 }  
 return clientArrayList;  
 }  
  
  
 public ArrayList<Client> getClientListSortByABC() {  
 ArrayList<Client> clientArrayList = new ArrayList<>(clientList);  
 Collections.*sort*(clientArrayList, new Comparator<Client>() {  
 @Override  
 public int compare(Client client1, Client client2) {  
 return client1.getClientSecondName().compareToIgnoreCase(client2.getClientSecondName());  
 }  
 });  
 return clientArrayList;  
 }  
}

Результат:

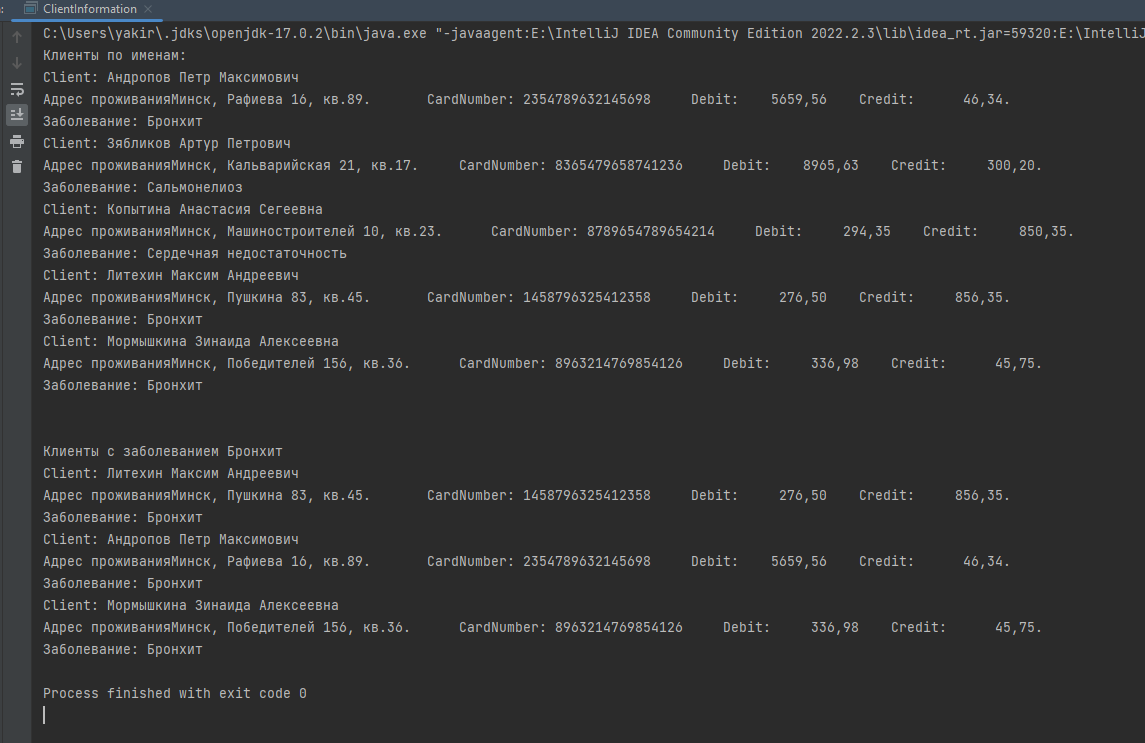


Рисунок – 2.

* 1. **Задача 3**

Создать объект класса Планета, используя классы Материк,Океан,Остров. Методы: вывести на консоль название материка, планеты, количество материков.

package com.company;  
  
public class Continent {  
 private String name;  
  
 public Continent(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

package com.company;  
  
public class Island {  
 private String name;  
  
 public Island(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

package com.company;  
  
public class Ocean {  
 private String name;  
  
 public Ocean(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

package com.company;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Planet {  
 private String name;  
  
 private List<Ocean> oceanList = new ArrayList<>();  
 private List<Continent> continentList = new ArrayList<>();  
 private List<Island> islandList = new ArrayList<>();  
  
 public Planet(String name) {  
 this.name = name;  
 }  
  
 public void addOcean(Ocean ocean) {  
 oceanList.add(ocean);  
 }  
  
 public void addContinent(Continent continent) {  
 continentList.add(continent);  
 }  
  
 public void addIsland(Island island) {  
 islandList.add(island);  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public List<Ocean> getOceanList() {  
 return oceanList;  
 }  
  
 public List<Continent> getContinentList() {  
 return continentList;  
 }  
  
 public List<Island> getIslandList() {  
 return islandList;  
 }  
}

package com.company;  
  
import java.util.List;  
  
public class PlanetRunner {  
 public static void main(String[] args) {  
 Island island = new Island("Barbados");  
  
 Planet planet = new Planet("Earth");  
  
 planet.addContinent(new Continent("Eurasia"));  
 planet.addContinent(new Continent("Africa"));  
  
 planet.addOcean(new Ocean("Atlantic"));  
 planet.addOcean(new Ocean("Pacific"));  
  
 planet.addIsland(new Island("Barbados"));  
  
 System.*out*.println("Planet name: "+planet.getName());  
 System.*out*.println("Continent name: "+*getFirstContinentName*(planet.getContinentList()));  
 System.*out*.println("Count continents:" + planet.getContinentList().size());  
 }  
  
 public static String getFirstContinentName(List<Continent> continentList) {  
 String result = null;  
 for (Continent continent: continentList) {  
 result = continent.getName();  
 }  
  
 return result;  
 }  
}

Результат:

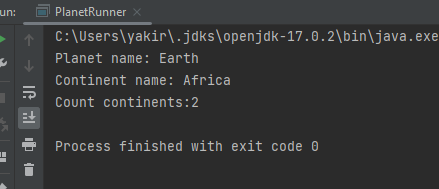


Рисунок – 3.

* 1. **Задача 4**

Создать класс Payment с внутренним классом, с помощью объектов которого можно сформировать покупку из нескольких товаров.

package com.company;  
  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
  
public class Task4 {  
  
  
 public static void main(String[] args) {  
 Account accaunt = new Account();  
  
 accaunt.newTransaction(Account.typeTransaction.*RECEIPT*, 1000);  
 accaunt.newTransaction(Account.typeTransaction.*PAYMENT*, 100);  
 accaunt.newTransaction(Account.typeTransaction.*WITHDRAWAL*, 1500);  
  
 System.*out*.println(accaunt.print());  
  
 }  
  
}  
  
  
class Account {  
  
 enum typeTransaction {  
 *WITHDRAWAL*, *PAYMENT*, *RECEIPT* }  
  
 int accountBalance = 0;  
 int nextId = 0;  
  
 List<Transaction> history = new ArrayList<>();  
  
 abstract class Transaction {  
 int id = 0;  
 int amount = 0;  
 int balance = 0;  
 Date date = new Date();  
 typeTransaction type;  
  
 public Transaction() {  
 this.id = nextId++;  
 }  
  
 @Override  
 public String toString() {  
 String string;  
 string = '\n' + "Transaction ID: " + this.id + '\n';  
 string += "Type: " + this.type + '\n';  
 string += "Date: " + this.date + '\n';  
 string += "Amount: " + this.amount + '\n';  
 string += "Balance after operation: " + this.balance + '\n';  
 return string;  
 }  
  
 }  
  
 class Withdrawal extends Transaction {  
  
 public Withdrawal(int amount) {  
 this.balance = accountBalance -= amount;  
 this.amount = amount;  
 }  
  
  
 }  
  
 class Payment extends Transaction {  
  
 public Payment(int amount) {  
 this.balance = accountBalance -= amount;  
 this.amount = amount;  
  
 }  
  
  
 }  
  
 class Receipt extends Transaction {  
  
 public Receipt(int amount) {  
 this.balance = accountBalance += amount;  
 this.amount = amount;  
 }  
  
  
 }  
  
 public void newTransaction(typeTransaction type, int amount)  
 throws UnsupportedOperationException {  
  
 Transaction newTransaction;  
  
 switch (type) {  
 case *WITHDRAWAL*:  
 newTransaction = new Withdrawal(amount);  
 break;  
  
 case *PAYMENT*:  
 newTransaction = new Payment(amount);  
 break;  
  
 case *RECEIPT*:  
 newTransaction = new Receipt(amount);  
 break;  
  
 default:  
 throw new UnsupportedOperationException("Not supported yet.");  
  
 }  
  
 newTransaction.type = type;  
 history.add(newTransaction);  
 }  
  
 public String print() {  
 String string = "";  
  
 for (Transaction t : history) {  
 string += t.toString();  
 }  
  
  
 return string;  
 }  
}

Результат:

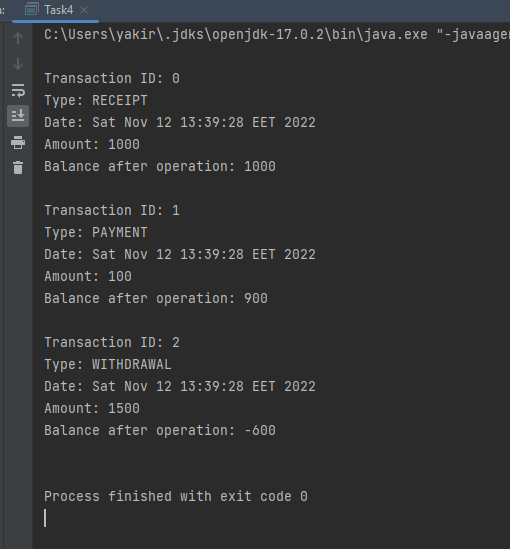


Рисунок – 4.

* 1. **Задача 5**

После каждого слова текста, заканчивающегося заданной подстрокой, вставить указанное слово.

package com.company;  
  
import java.util.Scanner;  
  
public class Task5 {  
  
 public static void main(String[] args) {  
 System.*out*.println("Введите окончание слова которое должно быть заменено: ");  
 Scanner s = new Scanner(System.*in*);  
 String suffix = s.nextLine

s.close();  
 String text = """  
 Two roads diverged in a yellow wood,  
 And sorry I could not travel both  
 And be one traveler, long I stood  
 And looked down one as far as I could  
 To where it bent in the undergrowth.""";  
 String string = "SecretWord";  
 String splited[] = text.split("\\s");

StringBuffer newText = new StringBuffer();  
 for (  
 int i = 0;  
 i < splited.length; i++) {  
 if (!splited[i].endsWith(suffix))

{  
 newText.append(splited[i] + " ");  
 } else {  
 newText.append(splited[i] + " " + string + " ");  
 }  
 }  
 System.*out*.println(newText);  
 }  
}

Результат:

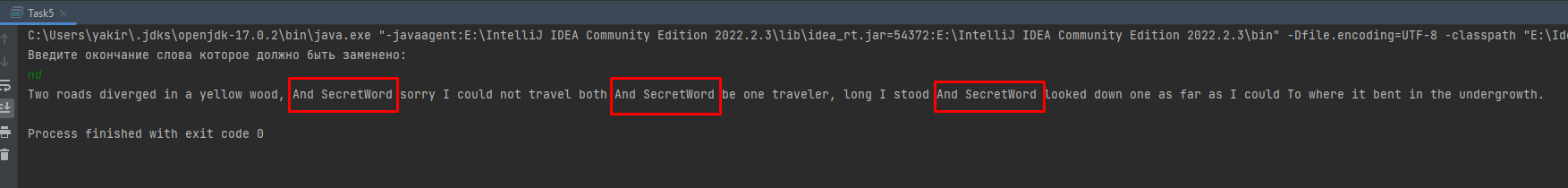


Рисунок – 5.

**1.6 Задача 6**

Вариант A

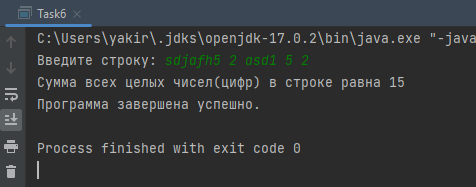
Выполнить задания на основе варианта А задачи 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т. д.

Вариант B

Выполнить задания из варианта В задачи 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

package com.company;  
  
import java.io.IOException;  
import java.text.NumberFormat;  
import java.util.Scanner;  
  
public class Task6 {  
  
 public static void main(String[] args) {  
 try {  
 System.*out*.print("Введите строку: ");  
 Scanner line = new Scanner(System.*in*);  
 String str = line.nextLine();  
 char[] chars = str.toCharArray();  
 int sum = 0;  
 for (int i = 0; i < chars.length; i++) {  
 if ((byte) chars[i] >= 0x30 && (byte) chars[i] <= 0x39)  
 sum += (byte) chars[i] - 0x30;  
 }  
 System.*out*.println("Сумма всех целых чисел(цифр) в строке равна " + sum);  
 }catch(NumberFormatException e) {  
 System.*err*.println(e.getMessage());  
 }finally{  
 System.*out*.println("Программа завершена успешно. ");  
 }  
 }  
}

Результат:



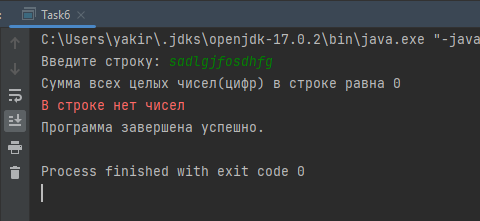


Рисунок – 6.

* 1. Разработка ER-диаграммы

Условие:



    В качестве серверной части разработать базу данных в СУБД MySQL в соответствии с вариантом. База данных содержит таблицу, состоящую из не менее чем 5-ти полей и 20-ти записей.

    Клиентская часть должна быть разработана на языке Java. Каждый запрос должен быть реализован отдельным методом. Все методы могут принадлежать одному классу. Отдельно должен быть реализован класс, содержащий метод main().

    Таким образом, структура клиента должна быть такой:

1. Класс, содержащий статический метод main() для демонстрации возможностей программы.
2. Класс, содержащий поля и методы для работы с удалённой базой данных:
3. Обязательные методы (запросы) клиента:

- выборка содержимого таблиц базы данных;

- запись в файл результата выборки содержимого таблиц БД.

- вывод в консоль результатов выполнения каждого метода.

1. Дополнительные методы (запросы) клиента
   1. добавление записи в таблицу.
   2. удаление записи из таблицы.
   3. обновление записи в таблице.
   4. поиск записи по признаку (на ваш выбор).

Техническое задание.

1) Анализ предметной области. Разработка ER-диаграммы «сущность-связь» (ErWin). Разработка диаграммы вариантов использования, диаграммы классов, диаграммы последовательностей, диаграммы кооперации, диаграммы состояний, диаграммы деятельности (MS Visio, Rational Rose и другие).

2) Разработать базу данных в СУБД MySQL «Театры г. Калининграда».

Клиентская часть должна быть разработана на языке Java. Каждый запрос должен быть реализован отдельным методом. Все методы могут принадлежать одному классу. Отдельно должен быть реализован класс, содержащий метод main().

3) Разработка структурной и функциональной схемы АИС.

4) Проектирование и реализация интерфейса.

5) Разработка встроенной системы справочного руководства и программного документа «Руководство пользователя».

6) Проведение тестирования.

