Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №3**

**Дисциплина: Платформо-независимое программирование**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К. А. Корнилов

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. И. Шиян

**Тема:** Разработка и использование интерфейсов.

**Цель**:

Ознакомиться с понятием интерфейса, правилами разработки, наследования и способами прикладного использования интерфейсов в Java.

**Задание:**

1. В вашем варианте задачи из ЛР 2 заменить базовый класс интерфейсом. Если позволяет задача, разбить его на два интерфейса.
2. Разработать UML – диаграмму и программу по созданной иерархии.
3. Написать программу с использованием коллекций для подсчета слов в строке.

**Ход работы:**

1. Вместо базового класса были написаны 3 интерфейса, содержащие списки методов, необходимые для выполнения

Код программы:

public interface ClientAccountAction {  
 public void addMoney(float summa);  
 public boolean subMoney(float summa);  
 public float check\_balance();  
  
}

public interface AccountHelpMethods {  
 public String toString();  
 public boolean equals(Object o);

public interface AccountAction{  
 public void closeAccount();  
 public void openAccount();  
  
}

1. Были написаны соответствующие реализации методов для каждого из классов иерархии

Код программы:

package javanew;  
  
public class DebtAccount implements ClientAccountAction,AccountHelpMethods,AccountAction{  
  
 private float percent;  
 private float creditTime;  
 private float balance;  
 private boolean open;  
  
 public DebtAccount(float percent,float creditTime){  
 this.percent=percent;  
 this.creditTime=creditTime;  
 this.balance = 0;  
 this.open = true;  
 }  
 public float getSumma(){  
 return this.balance;  
 }  
 public void setSumma(float summa){  
 this.balance=summa;  
 }  
  
 public void addMoney(float summa){  
 this.balance=this.balance+summa;  
 }  
 public boolean subMoney(float summa){  
 if(this.balance-summa>0){  
 this.balance=this.balance-summa;  
 return true;  
 }  
 return false;  
 }  
  
 public float check\_balance(){  
 if(this.open)return this.getSumma();  
 return 0;  
 }  
 public boolean getOpen(){  
 return this.open;  
 }  
 public void setOpen(boolean open){  
 this.open=open;  
 }  
  
 @Override  
 public void closeAccount() {  
 this.open=false;  
 }  
 public void openAccount() {  
 this.open=true;  
 }  
  
  
 public void setPercent(float percent){  
 this.percent=percent;  
 }  
 public float getPercent(){  
 return this.percent;  
 }  
  
 public void setCreditTime(float time){  
 this.creditTime=time;  
 }  
 public float getCreditTime(){  
 return this.creditTime;  
 }  
 @Override  
 public String toString(){  
 return super.toString()+" Percent:"+this.percent+" Credit Time:"+this.creditTime;  
 }  
 @Override  
 public boolean equals(Object o){  
 boolean result = false;  
 if(o!=null && o instanceof DebtAccount){  
 DebtAccount d = (DebtAccount) o;  
 if((this.getSumma()==d.getSumma())&&(this.getOpen()==d.getOpen())&&(this.getPercent()==d.getPercent())&&(this.getCreditTime()==d.getCreditTime())){  
 result=true;  
 }  
 }  
 return result;  
 }  
  
}

package javanew;  
  
public class SalaryAccount implements ClientAccountAction,AccountAction,AccountHelpMethods{  
 public SalaryAccount(){  
 this.open=false;  
 this.balance=0;  
 }  
 public void completeAutoPayments(){}  
  
 private float balance;  
 private boolean open;  
 public float getSumma(){  
 return this.balance;  
 }  
 public void setSumma(float summa){  
 this.balance=summa;  
 }  
 public void addMoney(float summa){  
 this.balance=this.balance+summa;  
 }  
 public boolean subMoney(float summa){  
 if(this.balance-summa>0){  
 this.balance=this.balance-summa;  
 return true;  
 }  
 return false;  
 }  
 public float check\_balance(){  
 if(this.open)return this.getSumma();  
 return 0;  
 }  
 public boolean getOpen(){  
 return this.open;  
 }  
 public void setOpen(boolean open){  
 this.open=open;  
 }  
  
 @Override  
 public void closeAccount() {  
 this.open=false;  
 }  
 public void openAccount() {  
 this.open=true;  
 }  
  
  
 @Override  
 public String toString(){  
 return "Balance: "+this.getSumma()+" Status:"+this.getOpen();  
 }  
 @Override  
 public boolean equals(Object o){  
 boolean result = false;  
 if(o!=null && o instanceof SalaryAccount){  
 SalaryAccount d = (SalaryAccount) o;  
 if((this.balance==d.getSumma())&&(this.open==d.getOpen())){  
 result=true;  
 }  
 }  
 return result;  
 }  
}

1. На основании заданий была составлена и написана иерархия классов.

Диаграмма классов полученной иерархии представлена на рисунке 1. Код создания классов представлен в приложении.

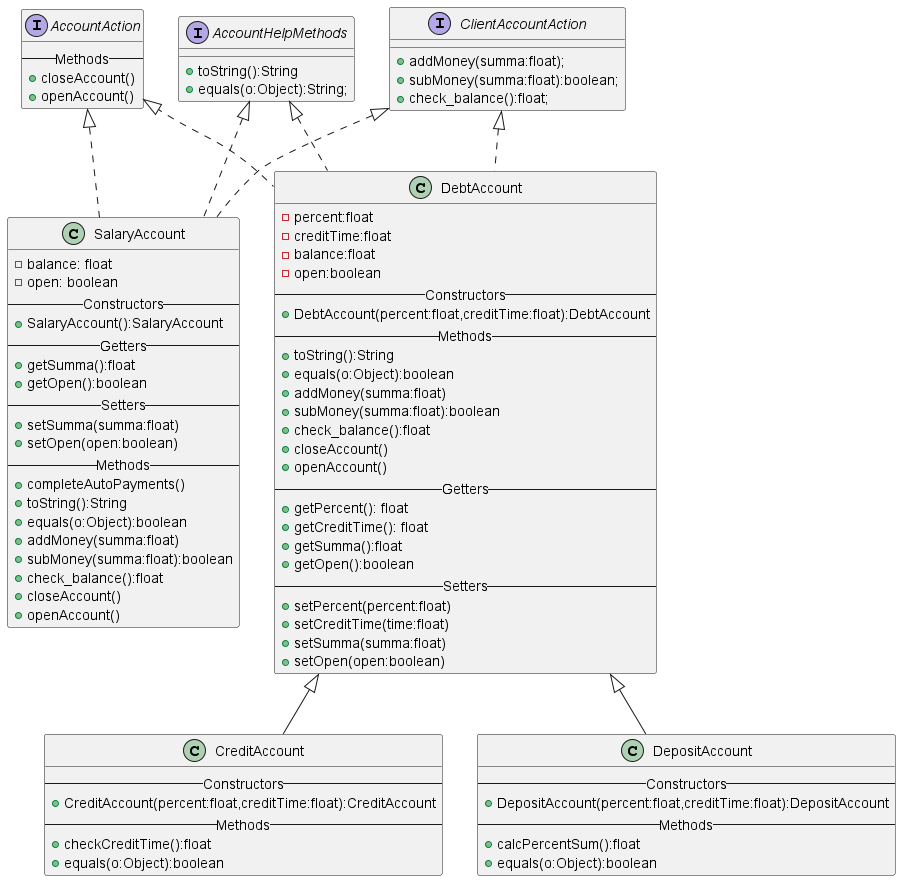


Рисунок 1 – Диаграмма классов иерархии.

1. Была написана программа с использованием коллекций для подсчета слов в строке.

Код программы:

public static void countOfWords(String inputString)  
{  
 HashMap<String,Integer> dictionary=new HashMap<>();  
 int i=0;  
 String buffer="";  
 inputString=inputString+' ';  
 while(i<inputString.length()){  
 if(Character.*isLetter*(inputString.charAt(i))){  
 buffer=buffer+inputString.charAt(i);  
 }  
 else if(buffer!=""){  
 int res = dictionary.getOrDefault(buffer,-1);  
 if(res==-1){  
 dictionary.put(buffer,1);  
 }  
 else{  
 dictionary.put(buffer,dictionary.get(buffer)+1);  
 }  
 buffer="";  
 }  
 i=i+1;  
 }  
 System.*out*.println(dictionary);  
}

Пример результата работы программы представлен на рисунке 2:



Рисунок 2 – Результат работы программы.

**Вывод:** Был освоен процесс построения интерфейсов на основе разработанных UML-диаграмм.

**ПРИЛОЖЕНИЕ**

**Полный код классов:**

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or  
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.  
package javanew;  
  
import java.util.HashMap;  
import java.util.Scanner;  
  
public class JavaNew {  
 public static void countOfWords(String inputString)  
 {  
 HashMap<String,Integer> dictionary=new HashMap<>();  
 int i=0;  
 String buffer="";  
 inputString=inputString+' ';  
 while(i<inputString.length()){  
 if(Character.*isLetter*(inputString.charAt(i))){  
 buffer=buffer+inputString.charAt(i);  
 }  
 else if(buffer!=""){  
 int res = dictionary.getOrDefault(buffer,-1);  
 if(res==-1){  
 dictionary.put(buffer,1);  
 }  
 else{  
 dictionary.put(buffer,dictionary.get(buffer)+1);  
 }  
 buffer="";  
 }  
 i=i+1;  
 }  
 System.*out*.println(dictionary);  
 }  
 public static void main(String[] args) {  
 Scanner in=new Scanner(System.*in*);  
 String inp = in.nextLine();  
 *countOfWords*(inp);  
 }  
}

package javanew;  
  
public class CreditAccount extends DebtAccount {  
 public CreditAccount(float percent,float creditTime){  
 super(percent,creditTime);  
 }  
 public float checkCreditTime(){  
 return this.getCreditTime();  
 }  
  
 @Override  
 public boolean equals(Object o){  
 return (o instanceof CreditAccount) && super.equals(o);  
 }  
}

package javanew;  
  
public class DebtAccount implements ClientAccountAction,AccountHelpMethods,AccountAction{  
  
 private float percent;  
 private float creditTime;  
 private float balance;  
 private boolean open;  
  
 public DebtAccount(float percent,float creditTime){  
 this.percent=percent;  
 this.creditTime=creditTime;  
 this.balance = 0;  
 this.open = true;  
 }  
 public float getSumma(){  
 return this.balance;  
 }  
 public void setSumma(float summa){  
 this.balance=summa;  
 }  
  
 public void addMoney(float summa){  
 this.balance=this.balance+summa;  
 }  
 public boolean subMoney(float summa){  
 if(this.balance-summa>0){  
 this.balance=this.balance-summa;  
 return true;  
 }  
 return false;  
 }  
  
 public float check\_balance(){  
 if(this.open)return this.getSumma();  
 return 0;  
 }  
 public boolean getOpen(){  
 return this.open;  
 }  
 public void setOpen(boolean open){  
 this.open=open;  
 }  
  
 @Override  
 public void closeAccount() {  
 this.open=false;  
 }  
 public void openAccount() {  
 this.open=true;  
 }  
  
  
 public void setPercent(float percent){  
 this.percent=percent;  
 }  
 public float getPercent(){  
 return this.percent;  
 }  
  
 public void setCreditTime(float time){  
 this.creditTime=time;  
 }  
 public float getCreditTime(){  
 return this.creditTime;  
 }  
 @Override  
 public String toString(){  
 return super.toString()+" Percent:"+this.percent+" Credit Time:"+this.creditTime;  
 }  
 @Override  
 public boolean equals(Object o){  
 boolean result = false;  
 if(o!=null && o instanceof DebtAccount){  
 DebtAccount d = (DebtAccount) o;  
 if((this.getSumma()==d.getSumma())&&(this.getOpen()==d.getOpen())&&(this.getPercent()==d.getPercent())&&(this.getCreditTime()==d.getCreditTime())){  
 result=true;  
 }  
 }  
 return result;  
 }  
  
}

package javanew;  
  
public class DepositAccount extends DebtAccount{  
 public DepositAccount(float percent,float creditTime){  
 super(percent,creditTime);  
 }  
 public float calcPercentSum(){  
 return this.getPercent();  
 }  
  
 @Override  
 public boolean equals(Object o){  
 return (o instanceof DepositAccount) && super.equals(o);  
 }  
}

package javanew;  
  
public class SalaryAccount implements ClientAccountAction,AccountAction,AccountHelpMethods{  
 public SalaryAccount(){  
 this.open=false;  
 this.balance=0;  
 }  
 public void completeAutoPayments(){}  
  
 private float balance;  
 private boolean open;  
 public float getSumma(){  
 return this.balance;  
 }  
 public void setSumma(float summa){  
 this.balance=summa;  
 }  
 public void addMoney(float summa){  
 this.balance=this.balance+summa;  
 }  
 public boolean subMoney(float summa){  
 if(this.balance-summa>0){  
 this.balance=this.balance-summa;  
 return true;  
 }  
 return false;  
 }  
 public float check\_balance(){  
 if(this.open)return this.getSumma();  
 return 0;  
 }  
 public boolean getOpen(){  
 return this.open;  
 }  
 public void setOpen(boolean open){  
 this.open=open;  
 }  
  
 @Override  
 public void closeAccount() {  
 this.open=false;  
 }  
 public void openAccount() {  
 this.open=true;  
 }  
  
  
 @Override  
 public String toString(){  
 return "Balance: "+this.getSumma()+" Status:"+this.getOpen();  
 }  
 @Override  
 public boolean equals(Object o){  
 boolean result = false;  
 if(o!=null && o instanceof SalaryAccount){  
 SalaryAccount d = (SalaryAccount) o;  
 if((this.balance==d.getSumma())&&(this.open==d.getOpen())){  
 result=true;  
 }  
 }  
 return result;  
 }  
}