**Міністерство освіти і науки України**

**Національний університет “Львівська політехніка”**

**Кафедра АСУ**

**Лабораторна робота №6**

Тема:

***«Функціонал програми»***

з курсу “Прикладне програмування”

Виконала:

Студентка групи КН-203

Трухим В.О.

Прийняв:

Вергун В.Р.

Львів 2020

**Завдання:**

Реалізуйте функціонал програми. Обов’язково зверніть увагу на додаткові вимоги до

програми (описані вище).

**18 варіант**

Формулювання задачі

В даній лабораторній роботі, основною задачею є - розробка програмного продукту за допомогою мови програмування JAVA.

Тема проекту – кредити.

*Сформувати набір пропозицій клієнту по цільових кредитах різних банків*

*для оптимального вибору. Враховувати можливість дострокового погашення*

*кредиту і/або збільшення кредитної лінії. Реалізувати вибір і пошук кредиту за*

*певними параметрами.*

Мета – вивчення нового матеріалу мови програмування JAVA та практика в написанні програми з використанням об’єктно-орієнтованого програмування.

**Виконання:**

**Main.java**

package Credits;  
  
public class Main {  
  
 public static void main(String[] args) {  
 MainMenu mainMenu = new MainMenu();  
 mainMenu.run();  
 }  
}

**MainMenu.java**

package Credits;  
  
import Credits.Menu.CommandMenu;  
import Credits.Menu.Commands.\*;  
import Credits.Model.Client;  
import Credits.Model.Credit.Credit;  
import Credits.Model.Credit.Type.Commercial;  
import Credits.Model.Credit.Type.Consumer;  
import Credits.Model.Credit.Type.Mortgage;  
import Credits.Settings.Bank;  
import Credits.Settings.CommandName;  
import Credits.Settings.MenuItem;  
import Credits.Settings.MenuItemName;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class MainMenu {  
 static boolean *creditChosen*;  
 private static final Logger *LOGGER* = Logger.*getLogger*( MainMenu.class.getName() );  
 private final ArrayList<Credit> credits;  
 private final Client client;  
 private final CommandMenu command;  
  
 public MainMenu() {  
 *creditChosen* = false;  
 credits = new ArrayList<>();  
 client = createNewClient();  
 command = new CommandMenu();  
  
 credits.add(new Commercial(13.6, 500000, 120, Bank.*ALFA*, true, true));  
 credits.add(new Consumer(17.2, 1000000, 240, Bank.*PRIVAT*, false, true));  
 credits.add(new Mortgage(10.1, 750000, 80, Bank.*MONO*, true, true));  
  
 command.setCommand(CommandName.*CHOOSE\_CREDIT\_COMMAND*.getName(), new ChooseCreditCommand());  
 command.setCommand(CommandName.*SHOW\_CHOSEN\_CREDIT\_COMMAND*.getName(), new ShowChosenCreditCommand());  
 command.setCommand(CommandName.*SHOW\_AVAILABLE\_CREDITS\_COMMAND*.getName(), new ShowAvailableCreditsCommand());  
 command.setCommand(CommandName.*INCREASE\_CREDIT\_LINE\_COMMAND*.getName(), new IncreaseCreditLineCommand());  
 command.setCommand(CommandName.*SEARCH\_CREDIT\_BY\_CRITERIA\_COMMAND*.getName(), new SearchCreditByCriteriaCommand());  
 command.setCommand(CommandName.*MAKE\_EARLY\_REPAYMENT\_COMMAND*.getName(), new MakeEarlyRepaymentCommand());  
 }  
  
 void run() {  
 *LOGGER*.log( Level.*FINE*, "run()" );  
 boolean quit = false;  
 while (!quit) {  
 showMenu();  
 switch(getMenuItem()) {  
 case *CHOOSE\_CREDIT*: command.runCommand(CommandName.*CHOOSE\_CREDIT\_COMMAND*.getName(), credits, client); break;  
 case *SHOW\_CHOSEN*: command.runCommand(CommandName.*SHOW\_CHOSEN\_CREDIT\_COMMAND*.getName(), credits, client); break;  
 case *SHOW\_AVAILABLE*: command.runCommand(CommandName.*SHOW\_AVAILABLE\_CREDITS\_COMMAND*.getName(), credits, client); break;  
 case *INCREASE\_CREDIT\_LINE*: command.runCommand(CommandName.*INCREASE\_CREDIT\_LINE\_COMMAND*.getName(), credits, client); break;  
 case *SEARCH\_CREDIT\_BY\_CRITERIA*: command.runCommand(CommandName.*SEARCH\_CREDIT\_BY\_CRITERIA\_COMMAND*.getName(), credits, client); break;  
 case *MAKE\_EARLY\_REPAYMENT*: command.runCommand(CommandName.*MAKE\_EARLY\_REPAYMENT\_COMMAND*.getName(), credits, client); break;  
 case *EXIT*: quit = true; break;  
 }  
 }  
 }  
  
 private void showMenu() {  
 System.*out*.print(MenuItemName.*CHOOSE\_CREDIT\_ITEM*.getName() +  
 MenuItemName.*SHOW\_CHOSEN\_CREDIT\_ITEM*.getName() +  
 MenuItemName.*SHOW\_AVAILABLE\_CREDITS\_ITEM*.getName() +  
 MenuItemName.*INCREASE\_CREDIT\_LINE\_ITEM*.getName() +  
 MenuItemName.*SEARCH\_CREDIT\_BY\_CRITERIA\_ITEM*.getName() +  
 MenuItemName.*MAKE\_EARLY\_REPAYMENT\_ITEM*.getName() +  
 MenuItemName.*EXIT*.getName());  
 }  
  
 private MenuItem getMenuItem() {  
 int item;  
 Scanner in = new Scanner(System.*in*);  
 do {  
 System.*out*.format("choose menu item -> ");  
 item = in.nextInt();  
 if (item == MenuItem.*CHOOSE\_CREDIT*.getValue()) {  
 System.*out*.println("Please choose credit first");  
 *creditChosen* = true;  
 }  
 } while(!*creditChosen* && item != MenuItem.*EXIT*.getValue() ||  
 item < MenuItem.*LOWER\_LIMIT*.getValue() ||  
 MenuItem.*UPPER\_LIMIT*.getValue() < item);  
  
 *LOGGER*.log(Level.*FINE*, "getMenuItem() returned %d menu item", item);  
 return MenuItem.*values*()[item];  
 }  
  
 private Client createNewClient() {  
 return new Client("Iryna", "Faryna", 1000000);  
 }  
}

**Bank.java**

package Credits.Settings;  
  
public enum Bank {  
 *ALFA*,  
 *PRIVAT*,  
 *MONO*}

**CommandName.java**

package Credits.Settings;  
  
public enum CommandName {  
  
 *CHOOSE\_CREDIT\_COMMAND*("ChooseCreditCommand"),  
 *SHOW\_CHOSEN\_CREDIT\_COMMAND*("ShowChosenCreditCommand"),  
 *SHOW\_AVAILABLE\_CREDITS\_COMMAND*("ShowAvailableCreditsCommand"),  
 *INCREASE\_CREDIT\_LINE\_COMMAND*("IncreaseCreditLineCommand"),  
 *SEARCH\_CREDIT\_BY\_CRITERIA\_COMMAND*("SearchCreditByCriteriaCommand"),  
 *MAKE\_EARLY\_REPAYMENT\_COMMAND*("MakeEarlyRepaymentCommand");  
  
 private final String name;  
  
 public String getName() {  
 return name;  
 }  
 CommandName(String name) {  
 this.name = name;  
 }  
  
}

**MenuItem.Java**

package Credits.Settings;  
  
public enum MenuItem {  
 *LOWER\_LIMIT*(1),  
 *CHOOSE\_CREDIT*(1),  
 *SHOW\_CHOSEN*(2),  
 *SHOW\_AVAILABLE*(3),  
 *INCREASE\_CREDIT\_LINE*(4),  
 *SEARCH\_CREDIT\_BY\_CRITERIA*(5),  
 *MAKE\_EARLY\_REPAYMENT*(6),  
 *EXIT*(7),  
 *UPPER\_LIMIT*(7);  
  
 private final int value;  
  
 MenuItem(int value) {  
 this.value = value;  
 }  
  
 public int getValue() {  
 return value;  
 }  
  
}

**MenuItemName.java**

package Credits.Settings;  
  
public enum MenuItemName {  
  
 *CHOOSE\_CREDIT\_ITEM*("1 - choose credit\n"),  
 *SHOW\_CHOSEN\_CREDIT\_ITEM*("2 - show chosen credit\n"),  
 *SHOW\_AVAILABLE\_CREDITS\_ITEM*("3 - show available credits\n"),  
 *INCREASE\_CREDIT\_LINE\_ITEM*("4 - increase credit line\n"),  
 *SEARCH\_CREDIT\_BY\_CRITERIA\_ITEM*("5 - search credit by criteria\n"),  
 *MAKE\_EARLY\_REPAYMENT\_ITEM*("6 - make early repayment\n"),  
 *EXIT*("7 - exit\n");  
  
 private final String name;  
  
 public String getName() {  
 return name;  
 }  
  
 MenuItemName(String name) {  
 this.name = name;  
 }  
}

**Credit.java**

package Credits.Model.Credit;  
  
import Credits.Settings.Bank;  
  
public abstract class Credit {  
 private final String name;  
 private final double rate;  
 private final double minSum;  
 private double duration;  
 private final Bank bank;  
 private final boolean possibilityOfLineIncrease;  
 private final boolean possibilityOfEarlyRepayment;  
  
 public Credit(String name, double rate, double minSum, double duration, Bank bank, boolean possibilityOfLineIncrease, boolean possibilityOfEarlyRepayment) {  
 this.name = name;  
 this.rate = rate;  
 this.minSum = minSum;  
 this.duration = duration;  
 this.bank = bank;  
 this.possibilityOfLineIncrease = possibilityOfLineIncrease;  
 this.possibilityOfEarlyRepayment = possibilityOfEarlyRepayment;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public double getRate() {  
 return rate;  
 }  
  
 public double getMinSum() {  
 return minSum;  
 }  
  
 public double getDuration() {  
 return duration;  
 }  
  
 public Bank getBank() {  
 return bank;  
 }  
  
 public boolean isPossibilityOfLineIncrease() {  
 return possibilityOfLineIncrease;  
 }  
  
 public boolean isPossibilityOfEarlyRepayment() {  
 return possibilityOfEarlyRepayment;  
 }  
  
 public void increaseCreditLine(int duration) {  
 if (possibilityOfLineIncrease)  
 this.duration += duration;  
 }  
  
 @Override  
 public String toString() {  
 return "Credit{" +  
 "name='" + name + '\'' +  
 ", rate=" + rate +  
 ", minSum=" + minSum +  
 ", duration=" + duration +  
 ", bank=" + bank +  
 ", possibilityOfLineIncrease=" + possibilityOfLineIncrease +  
 ", possibilityOfEarlyRepayment=" + possibilityOfEarlyRepayment +  
 '}';  
 }  
}

**Mortgage.java**

package Credits.Model.Credit.Type;  
  
import Credits.Model.Credit.Credit;  
import Credits.Settings.Bank;  
  
public class Mortgage extends Credit {  
 public Mortgage(double rate, double minSum, double duration, Bank bank, boolean possibilityOfLineIncrease, boolean possibilityOfEarlyRepayment) {  
 super("Mortgage", rate, minSum, duration, bank, possibilityOfLineIncrease, possibilityOfEarlyRepayment);  
 }  
}

**Consumer.java**

package Credits.Model.Credit.Type;  
  
import Credits.Model.Credit.Credit;  
import Credits.Settings.Bank;  
  
public class Consumer extends Credit {  
 public Consumer(double rate, double minSum, double duration, Bank bank, boolean possibilityOfLineIncrease, boolean possibilityOfEarlyRepayment) {  
 super("Consumer", rate, minSum, duration, bank, possibilityOfLineIncrease, possibilityOfEarlyRepayment);  
 }  
}

**Commercial.java**

package Credits.Model.Credit.Type;  
  
import Credits.Model.Credit.Credit;  
import Credits.Settings.Bank;  
  
public class Commercial extends Credit {  
 public Commercial(double rate, double minSum, double duration, Bank bank, boolean possibilityOfLineIncrease, boolean possibilityOfEarlyRepayment) {  
 super("Commercial", rate, minSum, duration, bank, possibilityOfLineIncrease, possibilityOfEarlyRepayment);  
 }  
}

**Client.java**

package Credits.Model;  
  
import Credits.Model.Credit.Credit;  
  
public class Client {  
 private final String firstName;  
 private final String lastName;  
 private double money;  
 private Credit credit;  
  
 public Client(String firstName, String lastName, double money) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.money = money;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public double getMoney() {  
 return money;  
 }  
  
 public Credit getCredit() {  
 return credit;  
 }  
  
  
 public void chooseCredit(Credit credit) {  
 this.credit = credit;  
 }  
  
 public void makeEarlyRepayment(int money) {  
 if (credit.isPossibilityOfEarlyRepayment())  
 this.money -= money;  
 }  
  
 @Override  
 public String toString() {  
 return "Client{" +  
 "firstName='" + firstName + '\'' +  
 ", lastName='" + lastName + '\'' +  
 ", money=" + money +  
 ", credit=" + credit +  
 '}';  
 }  
}

**Command.java**

package Credits.Menu.Commands;  
  
import Credits.Model.Client;  
import Credits.Model.Credit.Credit;  
  
import java.util.ArrayList;  
  
public interface Command {  
 void execute(ArrayList<Credit> credit, Client client);  
}

**ChooseCreditCommand.java**

package Credits.Menu.Commands;  
  
import Credits.Model.Client;  
import Credits.Model.Credit.Credit;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class ChooseCreditCommand implements Command {  
 private static final Logger *LOGGER* = Logger.*getLogger*( ChooseCreditCommand.class.getName() );  
 @Override  
 public void execute(ArrayList<Credit> credits, Client client) {  
 *LOGGER*.log( Level.*FINE*, "execute()" );  
 showAvailableCredits(credits);  
 client.chooseCredit(getChosenCredit(credits));  
 }  
  
 private void showAvailableCredits(ArrayList<Credit> credits) {  
 *LOGGER*.log( Level.*FINE*, "showAvailableCredits()" );  
 for (Credit credit: credits) {  
 System.*out*.println(credit);  
 }  
 }  
  
 private Credit getChosenCredit(ArrayList<Credit> credits) {  
 *LOGGER*.log( Level.*FINE*, "getChosenCredit()" );  
 System.*out*.println("choose number of credit you want to get");  
 Scanner in = new Scanner(System.*in*);  
 int creditNumber = in.nextInt();  
  
 return credits.get(creditNumber);  
 }  
  
}

**IncreaseCreditLineCommand.java**

package Credits.Menu.Commands;  
  
import Credits.Model.Client;  
import Credits.Model.Credit.Credit;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class IncreaseCreditLineCommand implements Command {  
 private static final Logger *LOGGER* = Logger.*getLogger*( IncreaseCreditLineCommand.class.getName() );  
 @Override  
 public void execute(ArrayList<Credit> credit, Client client) {  
 *LOGGER*.log(Level.*FINE*, "execute()");  
 if (client.getCredit().isPossibilityOfLineIncrease()) {  
 System.*out*.print("enter for how many months you want increase credit line: ");  
 Scanner in = new Scanner(System.*in*);  
 int creditNumber = in.nextInt();  
 client.getCredit().increaseCreditLine(creditNumber);  
 }  
 else {  
 System.*out*.print("Your credit has no option to increase credit line");  
 }  
 }  
}

**MakeEarlyRepaymentCommand.java**

package Credits.Menu.Commands;  
  
import Credits.Model.Client;  
import Credits.Model.Credit.Credit;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class MakeEarlyRepaymentCommand implements Command {  
 private static final Logger *LOGGER* = Logger.*getLogger*( MakeEarlyRepaymentCommand.class.getName() );  
  
 @Override  
 public void execute(ArrayList<Credit> credits, Client client) {  
 *LOGGER*.log( Level.*FINE*, "execute()" );  
 if (client.getCredit().isPossibilityOfEarlyRepayment()) {  
 System.*out*.print("enter for how many you want to repay the credit: ");  
 Scanner in = new Scanner(System.*in*);  
 int money = in.nextInt();  
 client.makeEarlyRepayment(money);  
 }  
 else {  
 System.*out*.print("Your credit has no option to make early repayment");  
 }  
 }  
  
}

**SearchCreditByCriteriaCommand.java**

package Credits.Menu.Commands;  
  
import Credits.Model.Client;  
import Credits.Model.Credit.Credit;  
  
import java.util.ArrayList;  
import java.util.InputMismatchException;  
import java.util.Scanner;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class SearchCreditByCriteriaCommand implements Command {  
 private static final Logger *LOGGER* = Logger.*getLogger*( SearchCreditByCriteriaCommand.class.getName() );  
  
 @Override  
 public void execute(ArrayList<Credit> credits, Client client) {  
 *LOGGER*.log( Level.*FINE*, "execute()" );  
 search(credits);  
 }  
  
 private void search(ArrayList<Credit> credits) {  
 *LOGGER*.log( Level.*FINE*, "search()" );  
 try {  
 boolean foundRecords = false;  
 switch (getCriteria()) {  
 case 1: foundRecords = searchByLineIncrease(credits); break;  
 case 2: foundRecords = searchByRepayment(credits); break;  
 }  
  
 if (!foundRecords)  
 System.*out*.println("There are no such credits");  
 }  
 catch (InputMismatchException ex) {  
 *LOGGER*.log( Level.*SEVERE*, ex.toString(), ex );  
 System.*out*.println("Bad input!!!!");  
 }  
 }  
  
 private int getCriteria() {  
 *LOGGER*.log( Level.*FINE*, "getCriteria()" );  
 System.*out*.println("1 - search by possibility of credit line increase\n" +  
 "2 - search by possibility of early repayment ");  
 Scanner in = new Scanner(System.*in*);  
 return in.nextInt();  
 }  
  
  
 private boolean searchByLineIncrease (ArrayList<Credit> credits) {  
 *LOGGER*.log( Level.*FINE*, "searchByLineIncrease()" );  
 boolean foundRecords = false;  
 for (Credit credit: credits) {  
 if (credit.isPossibilityOfLineIncrease()) {  
 System.*out*.println(credit);  
 foundRecords = true;  
 }  
 }  
 return foundRecords;  
 }  
  
 private boolean searchByRepayment (ArrayList<Credit> credits) {  
 *LOGGER*.log( Level.*FINE*, "searchByRepayment()" );  
 boolean foundRecords = false;  
 for (Credit credit: credits) {  
 if (credit.isPossibilityOfEarlyRepayment()) {  
 System.*out*.println(credit);  
 foundRecords = true;  
 }  
 }  
 return foundRecords;  
 }  
}

**ShowAvailableCreditsCommand.java**

package Credits.Menu.Commands;  
  
import Credits.Model.Client;  
import Credits.Model.Credit.Credit;  
  
import java.util.ArrayList;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class ShowAvailableCreditsCommand implements Command {  
 private static final Logger *LOGGER* = Logger.*getLogger*( ShowAvailableCreditsCommand.class.getName() );  
  
 @Override  
 public void execute(ArrayList<Credit> credits, Client client) {  
 *LOGGER*.log( Level.*FINE*, "execute()" );  
 for (Credit credit: credits) {  
 System.*out*.println(credit);  
 }  
 }  
}

**ShowChosenCreditCommand.java**

package Credits.Menu.Commands;  
  
import Credits.Model.Client;  
import Credits.Model.Credit.Credit;  
  
import java.util.ArrayList;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class ShowChosenCreditCommand implements Command {  
 private static final Logger *LOGGER* = Logger.*getLogger*(ShowChosenCreditCommand.class.getName());  
  
 @Override  
 public void execute(ArrayList<Credit> credit, Client client) {  
 *LOGGER*.log(Level.*FINE*, "execute()");  
 System.*out*.println("Your credit is " + client.getCredit());  
 }  
}

**CommandMenu.java**

package Credits.Menu;  
  
import Credits.Menu.Commands.Command;  
import Credits.Model.Client;  
import Credits.Model.Credit.Credit;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class CommandMenu {  
 private static final Logger *LOGGER* = Logger.*getLogger*( CommandMenu.class.getName() );  
 Map<String, Command> menuItems;  
  
 public CommandMenu() {  
 menuItems = new HashMap<String, Command>();  
 }  
  
 public void setCommand(String operation, Command cmd) {  
 menuItems.put(operation, cmd);  
 *LOGGER*.log( Level.*FINE*, "setCommand() created %s command", operation );  
 }  
  
 public void runCommand(String operation, ArrayList<Credit> credit, Client client) {  
 menuItems.get(operation).execute(credit, client);  
 *LOGGER*.log( Level.*FINE*, "runCommand() executed %s command", operation );  
 }  
}