Основи технологій програмування

Лабораторна робота № 5

Вінницький В’ячеслав Андрійович

ІП-64, 2-ий курс

Кафедра обчислювальної техніки

ІП-6402

Варіант

С3 = 6402%3 = 0 => Тип StringBuilder

C17 = 6402%17 = 10 => З кожного речення заданого тексту видалити підрядок найбільшої довєини що починається та закінчуєтьсяи заданими літерами

Текст програмного коду

import java.util.Scanner;  
/\*  
завдання полягає в наступному: с17 = 7116 % 17 = 10  
Сформувати набір пропозицій клієнту по цільовим кредитам різних банків.  
Враховувати можливість дострокового погашення кредиту й\або збільшення  
кредитної лінії. Реалізувати вибір та пошук кредиту за будь-якими  
параметрами.  
 \*/  
public class Main {  
 public static void main(String[] args) {  
 int chooseOrFind;  
 int check\_bank;  
 int credit;  
 int suma;  
 int period;  
 int addAction;  
 int new\_period;  
 float newCreditMany;  
 /\* Старт програми відбувається тут  
 Створюється масив об'єктів тьрох банків з якими ми можемо виконувати надалі дії у змінній creditAction  
 \*/  
 CreditAction creditAction = new CreditAction(3);  
 creditAction.addNewBank(new Bank1(43.2f, 17.9f, 18.0f, 60, 120));  
 creditAction.addNewBank(new Bank2(55.6f, 19.9f, 22.0f, 48, 180));  
 creditAction.addNewBank(new Bank3(49.9f, 19.9f, 19.9f, 72, 240));  
 /\*  
 Інтерфейс програми, змінна "і" та "j" керує виходом з програми  
 \*/  
 boolean i = true;  
 boolean j;  
 while (i){  
 j = true;  
 System.*out*.println("1--Вибір кредиту за банком\n2--Пошук кредиту за параметрами\n3--Вихід");  
 chooseOrFind = *ChooseAction*(3);  
 while (j){  
 if (chooseOrFind == 1) {  
 //ввід від користувача, вибір банку  
 System.*out*.println("1--Банк 1\n2--Банк 2\n3--Банк 3");  
 check\_bank = *ChooseAction*(3);  
 //ввід від користувача, вибір цільового кредиту  
 System.*out*.println("1--Звичайний кредит\n2--Іпотека\n3--Кредит на авто");  
 credit = *ChooseAction*(3);  
 //ввід від користувача, сума кредиту  
 System.*out*.println("Введіть суму кредиту - ");  
 suma = *ChooseAction*(500000);  
 //ввід від користувача, період кредиту  
 System.*out*.println("На який період - ");  
 period = *ChooseAction*(creditAction.getMaxMonthsCredit(check\_bank, credit));  
 // у змінній creditMany зберігається сума яку потрібно віддати банку за кредитний період  
 float creditMany = creditAction.getNewCredit(check\_bank, credit, suma, period);  
 System.*out*.printf("Взятий новий кредит на суму %d грн на період %d місяці\nПотрібно повернути %f грн\n",  
 suma , period, creditMany);  
 System.*out*.println("1--Достроково погасити\n2--Збільшити період\n3--Нові кредити\n4--Вихід");  
 //ввід від користувача, вибір способу взаємодії1  
 addAction = *ChooseAction*(4);  
 if (addAction == 4) {  
 i = false;  
 j = false;  
 }else if (addAction == 3) {  
 i = true;  
 j = false;  
 }else if (addAction == 2) {  
 // тут знаходиться блок збільшення часу на поврнення кредиту  
 System.*out*.println("Вкажіть новий період - ");  
 new\_period = *ChooseAction*(creditAction.getMaxMonthsCredit(check\_bank, credit));  
 newCreditMany = *newPeriod*(check\_bank, credit, suma, new\_period, creditAction);  
 System.*out*.printf("Треба було повернути %f грн\nА тепер треба повернути %f грн\n",  
 creditMany, newCreditMany);  
 creditMany = newCreditMany;  
 period = new\_period;  
 } else {  
 // дострокове погашення кредиту  
 System.*out*.printf("Достроково погасити %f грн?\n1--Так\n2--Ні\n", creditMany);  
 if (*ChooseAction*(2) == 1)  
 System.*out*.printf("Кредит %f грн достроково погашено, заплачено було %d\n", creditMany, suma);  
 else  
 j = false;  
 }  
  
 }  
 else if (chooseOrFind == 2){  
 // пошук по деяким пареметрам, всі функції в класі CreditAction  
 System.*out*.println("1--Найменший та найбільший відсоток\n2--Найдовший та найменший період\n3--До головного меню");  
 credit = *ChooseAction*(3);  
 if (credit == 3)  
 j = false;  
 else if(credit == 2){  
 System.*out*.printf("Найдовший період - %d\nНайкоротший - %d\n",  
 creditAction.MinMaxMonths(0), creditAction.MinMaxMonths(5));  
 }else  
 System.*out*.printf("Найдовший період - %f\nНайкоротший - %f\n",  
 creditAction.MinMaxCredit(0), creditAction.MinMaxCredit(5));  
  
 }  
 else {  
 j = false;  
 i = false;  
 }  
 }  
 }  
 }  
 //метод для вибору дії в інтерфейсі  
 private static int ChooseAction(int max){  
 int num;  
 Scanner scan = new Scanner(System.*in*);  
 do {  
 System.*out*.printf("Ведіть число від %d до %d: ", 1, max);  
 while (!scan.hasNextInt()) {  
 System.*out*.printf("Ведіть число!!! від %d до %d: ", 1, max);  
 scan.next();  
 }  
 num = scan.nextInt();  
 }while ((num > max) || (num < 1));  
 return num;  
 }  
 // метод створення нового періоду, використовується метод getNewCredit як і для звичайного кредиту  
 private static float newPeriod(int bank, int credit, int suma, int new\_period, CreditAction creditAction){  
 return creditAction.getNewCredit(bank, credit, suma, new\_period);  
 }  
  
}

public class MainBank {  
 private int minMonths, maxMonthsUsual, maxMonthsCarAndHome;  
 private float percentUsualCredit, percentHomeCredit, percentCarCredit;  
 //в конструкторі приймаються значення процентів звичайного, іпотечного та кредиту на автомобіль  
 //та максимально можливий термін взяття кредиту  
 MainBank(float percentUsualCredit, float percentHomeCredit, float percentCarCredit,  
 int maxMonthsUsual, int maxMonthsCarAndHome) {  
 this.percentCarCredit = percentCarCredit;  
 this.percentHomeCredit = percentHomeCredit;  
 this.percentUsualCredit = percentUsualCredit;  
 this.maxMonthsUsual = maxMonthsUsual;  
 this.maxMonthsCarAndHome = maxMonthsCarAndHome;  
 }  
 //гетери для отримання значеннь відсотків та макс значень  
 public int getMaxMonthsUsual() {  
 return maxMonthsUsual;  
 }  
  
 public int getMaxMonthsCarAndHome() {  
 return maxMonthsCarAndHome;  
 }  
  
 public float getPercentUsualCredit() {  
 return percentUsualCredit;  
 }  
  
 public float getPercentHomeCredit() {  
 return percentHomeCredit;  
 }  
  
 public float getPercentCarCredit() {  
 return percentCarCredit;  
 }  
}

class Bank1 extends MainBank{  
 //просто банк який наслідує головний банк  
 Bank1(float percentUsualCredit, float percentHomeCredit, float percentCarCredit,  
 int maxMonthsUsual, int maxMonthsCarAndHome) {  
 super(percentUsualCredit, percentHomeCredit, percentCarCredit,  
 maxMonthsUsual, maxMonthsCarAndHome);  
 }  
}

class Bank2 extends MainBank{  
 //ще один банк  
 Bank2(float percentUsualCredit, float percentHomeCredit, float percentCarCredit,  
 int maxMonthsUsual, int maxMonthsCarAndHome) {  
 super(percentUsualCredit, percentHomeCredit, percentCarCredit,  
 maxMonthsUsual, maxMonthsCarAndHome);  
 }  
}

class Bank3 extends MainBank{  
 //і ще один банк  
 Bank3(float percentUsualCredit, float percentHomeCredit, float percentCarCredit,  
 int maxMonthsUsual, int maxMonthsCarAndHome) {  
 super(percentUsualCredit, percentHomeCredit, percentCarCredit,  
 maxMonthsUsual, maxMonthsCarAndHome);  
 }  
}

import java.util.Arrays;  
  
public class CreditAction {  
 private MainBank[] mainBanks;  
 private int numberOfBank = 0;  
 //в конструкторі вказуєм розмірність масиву об'єктів  
 CreditAction(int numberOfBank) {  
 mainBanks = new MainBank[numberOfBank];  
 }  
 // заповнюєм масив об'єктів екземплярами класів нащадків  
 public void addNewBank(MainBank mainBank) {  
 mainBanks[numberOfBank] = mainBank;  
 numberOfBank++;  
 }  
 // в цьому методі "берем новий" кредит  
 public float getNewCredit(int bank, int credit, int suma, int period) {  
 float percentCredit = 0.0f;  
 switch (credit) {  
 case 1:  
 percentCredit = mainBanks[bank - 1].getPercentUsualCredit();  
 break;  
 case 2:  
 percentCredit = mainBanks[bank - 1].getPercentHomeCredit();  
 break;  
 case 3:  
 percentCredit = mainBanks[bank - 1].getPercentCarCredit();  
 break;  
 }  
 return suma \* (1 + (percentCredit / 100) \* (period / 12));  
 }  
 // дізнаємось максимальний період на який можна взяти кредит  
 public int getMaxMonthsCredit(int bank, int credit) {  
 int a = 0;  
 switch (credit) {  
 case 1:  
 a = mainBanks[bank - 1].getMaxMonthsUsual();  
 break;  
 case 2:  
 a = mainBanks[bank - 1].getMaxMonthsCarAndHome();  
 break;  
 case 3:  
 a = mainBanks[bank - 1].getMaxMonthsCarAndHome();  
 break;  
 }  
 return a;  
 }  
 //повертаються значення мінімальних та максимальних відсотків і місяців  
 public int MinMaxMonths(int who) {  
 int[] creditMonths = new int[6];  
 int i = 0;  
 for (MainBank mainBank : mainBanks) {  
 creditMonths[i] = mainBank.getMaxMonthsUsual();  
 creditMonths[i + 1] = mainBank.getMaxMonthsCarAndHome();  
 i += 2;  
 }  
 System.*out*.println(Arrays.*toString*(creditMonths));  
 int flag;  
 for (int j = 1; j < creditMonths.length; j++) {  
 for (int k = creditMonths.length - 1; k >= j; k--) {  
 if (creditMonths[k] > creditMonths[k - 1]) {  
 flag = creditMonths[k - 1];  
 creditMonths[k - 1] = creditMonths[k];  
 creditMonths[k] = flag;  
 }  
 }  
 }  
  
 System.*out*.println(Arrays.*toString*(creditMonths));  
 if (who == 0)  
 return creditMonths[who];  
 else  
 return creditMonths[who];  
 }  
  
 public float MinMaxCredit(int who) {  
 float[] creditCredit = new float[9];  
 int i = 0;  
 for (MainBank mainBank : mainBanks) {  
 creditCredit[i] = mainBank.getPercentCarCredit();  
 creditCredit[i + 1] = mainBank.getPercentHomeCredit();  
 creditCredit[i + 2] = mainBank.getMaxMonthsUsual();  
 i += 3;  
 }  
 float flag;  
 for (int j = 1; j < creditCredit.length; j++) {  
 for (int k = creditCredit.length - 1; k >= j; k--) {  
 if (creditCredit[k] > creditCredit[k - 1]) {  
 flag = creditCredit[k - 1];  
 creditCredit[k - 1] = creditCredit[k];  
 creditCredit[k] = flag;  
 }  
 }  
 }  
 if (who == 0)  
 return creditCredit[who];  
 else  
 return creditCredit[creditCredit.length - 1];  
 }  
}

Діаграма класів

