**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

*Факультет прикладної математики*

*Кафедра «Спеціалізовані комп’ютерні системи»*

Лабораторна робота №2

з дисципліни

«Операційні системи»

**Виконав:** Шулик В.В. гр. КВ-81

м. Київ

2011

**Вариант №5**

**Модель автомата продажи билетов**

Автомат принимает денежные купюры (в данной реализации купюру номиналом в 1 гривню) и выдаёт сдачу мелочью в пределах одной гривни. Монеты достоинством в 1,2,5,10,25 и 50 копеек. Сумма сдачи **х** – случайная величина. Первоначальное количество монет каждого номинала, которыми выдаётся сдача, задано (1 коп. – 50 шт., 2 коп – 25 шт., 5 коп. – 20 шт., 10 коп. – 15 шт., 25 коп. – 10 шт., 50 коп. – 5 шт.). Ввод запроса на продажу организован следующим образом. Вводится число от 0 до 5:

* 0 – выключить автомат;
* 1 – купить билет в Киев, его стоимость – 28 коп.;
* 2 – купить билет в Москву, его стоимость – 37 коп.;
* 3 – купить билет в Лондон, его стоимость – 50 коп.;
* 4 – купить билет в Берлин, его стоимость – 77 коп.;
* 5 – купить билет в Париж, его стоимость – 91 коп.

Если сдачу выдать возможно, программа уменьшает соответствующие количества монет каждого из достоинств, которыми выдаётся сдача и формирует сигнал на выдачу сдачи. В противном случае, если оставшимися в наличии монетами сдачу выдать невозможно, формируется сигнал о невозможности продажи билета.

Требования на продажу билетов поступают через случайные промежутки времени после очередного сеанса продажи (или отказа от продажи) билета.

Автомат имеет 2 терминала («А» и «В»), которые работают независимо.

Модель автомата представить в виде трёх взаимодействующих процессов А, В, и С.

Процессы А и В определяют моменты поступления требования на продажу, требуемую величину сдачи.

Процесс С ожидает момент появления необходимости выдать сдачу и, если величина сдачи известна, рассчитывает количества монет каждого достоинства для выдачи сдачи и остаток или определяет невозможность её выдачи, о чем формирует соответствующее сообщение.

Разработать программу работы автомата, используя монитор Хоара для организации доступа к разделяемым ресурсам.

**Код програми**

**Main.java**

package os2;

public class Main {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

int delayA = 300;

int delayB = 100;

/\*

\* 1: білет за 28 коп.;

\* 2: білет за 37 коп.;

\* 3: білет за 50 коп.;

\* 4: білет за 77 коп.;

\* 5: білет за 91 коп.;

\*/

int tasksA[] = {1, 5, 8, 1};

int tasksB[] = {4, 3, 4,1,1,1,1,1,1,1};

int countT = tasksA.length + tasksB.length;

System.out.println("ПОЧАТОК РОБОТИ:\n");

Common sM = new Common('W');

ThreadC C = new ThreadC('C', sM, countT);

ThreadAB A = new ThreadAB(tasksA, 'A', delayA, sM, delayA);

ThreadAB B = new ThreadAB(tasksB, 'B', delayB, sM, delayB);

Thread tC = new Thread(C);

Thread tA = new Thread(A);

Thread tB = new Thread(B);

tC.start();

tA.start();

tB.start();

try {

tC.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("КІНЕЦЬ РОБОТИ");

}

}

**ThreadAB.java**

package os2;

public class ThreadAB implements Runnable{

char name;

Common sM;

int i = 20;

int tasks[];

int delay;

int maxD;

ThreadAB(int tasks[], char name, int delay, Common sM, int maxD){

this.name = name;

this.sM = sM;

this.tasks = new int[tasks.length];

this.maxD = maxD;

for(int i = 0; i < tasks.length; i++)

this.tasks[i] = tasks[i];

}

//RETURN COST OF TICKET

private int getTicket(int task){

switch(task)

{

case 1: return 28;

case 2: return 37;

case 3: return 50;

case 4: return 77;

case 5: return 91;

}

return 0;

}

//COUNT CHANGE FROM TICKET

private int countChange(int ticketId){

int price = getTicket(ticketId);

if (price == 0)

return 0;

return 100 - price;

}

public void run(){

int change ;

for(int i = 0; i < tasks.length; i++)

{

change = countChange(tasks[i]);

if (maxD < 100) maxD = 100;

sM.put(change, tasks[i], 100 - change, name, maxD);

try {

Thread.sleep(500);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

**ThreadC.java**

package os2;

public class ThreadC implements Runnable{

char name;

Common sM;

int num;

ThreadC(char name, Common sM, int count){

this.name = name;

this.sM = sM;

num = count;

}

public void run(){

while(num != 0) {

sM.work();

num--;

}

}

}

**Common.java**

package os2;

import java.util.Random;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Common {

/\*\*

\* COINS - CONTAIN ALL COINS

\* GIVING - CONTAIN POINTER TO CELLS OF COINS, WHICH WAS DECRIMENT

\* gNUM - NUMBER OF GIVING COINS

\* uNum - NUMBER OF UNIQUE GIVING COINS

\* uPos - POSITION OF NEXT UNIQUE COIN

\*

\* @author McDre

\*/

int[][] coins = {{50, 25, 10, 5, 2, 1}, {1, 10, 15, 20, 25, 50}};

int randD;

int delay;

int task;

int price;

char name;

int change = 0;

boolean set = false;

Common(char name){

this.name = name;

}

synchronized void put(int change, int task, int price, char name, int maxD){

delay = maxD;

if (set)

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Процес "+name+" почав завдання "+task+".");

this.change = change;

this.task = task;

this.price = price;

System.out.println(" Процес "+name+" сформував здачу у кількості "+change+" коп.");

System.out.println("Процес "+name+" закінчив завдання "+task+".");

System.out.println("");

set = true;

notify();

del();

}

synchronized void work(){

if (!set)

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

count(change, name, task, price, 1000);

set = false;

notify();

}

//COUNT BANK

private int bank(){

int bank = 0;

for(int i = 0; i < coins[0].length; i++)

bank += coins[0][i] \* coins[1][i];

return bank;

}

//Dec COINS IN BANK

private void giveCoins(int[] giving, int gNum){

for(int i = 0; i < gNum; i++){

this.coins[1][giving[i]]--;

}

}

//SHOW BANK

private void showBank(){

System.out.println(" Банк:");

int j = 0;

System.out.print(" номінал: ");

for(j=0; j < 2; j++){

if (j == 1) System.out.print("к-сть : ");

for(int i = 0; i < coins[0].length; i++){

System.out.print(coins[j][i]+" ");

if(coins[j][i] < 10) System.out.print(" ");

}

if (j == 1) System.out.println(" сума монет:" + bank());

System.out.println("");

System.out.print(" ");

}

}

//CREATE VIRTUAL BANK

private int[] virtualNum(int[][] coins){

int[] virt = new int[coins[0].length];

for(int i = 0; i < coins[0].length; i++)

virt[i] = coins[1][i];

return virt;

}

//COUNT NUMBER OF USED COINS

private int num(int coin, int[] giving){

int num = 0, i;

for(i = 0; i < this.coins[0].length; i++)

if (this.coins[0][i] == coin) {

coin = i;

break;

}

for(i = 0; i < giving.length; i++)

if (giving[i] == coin) {

if (i != 0 && coin == 0) return num;

num++;

if (i == 120 || giving[i+1] != coin) return num;

}

return num;

}

//MAKE RANDOM

private int rand(int max){

Random r = new Random();

int res = (int) (max \* r.nextDouble());

return res;

}

//DELAY

private void del(){

randD = rand(delay);

try {

Thread.sleep(randD);

} catch (InterruptedException ex) {

Logger.getLogger(ThreadAB.class.getName()).log(Level.SEVERE, null, ex);

}

}

//FINISH SOUT

private void finish (){

System.out.println("Процес C закінчив завдання " + task+ ". Затримка - " + randD + "\n");

}

//MAIN METHOD

private int count(int change, char name, int task, int price, int delay){

int giving[] = new int[121];

int gNum = 0, uNum = 0, uPos = 0;

int result = change;

this.delay = delay;

System.out.println("Процес C почав завдання " + task);

showBank();

if (change >= 100){

System.out.println("Здача неможлива. Розрахована здача дорівнює гриані або більша за неї.\n");

randD = 0;

finish();

return 0;

}

if (change <= 0){

System.out.println("Здача неможлива. Розрахована здача дорівнює нулю або від'ємна.\n");

randD = 0;

finish();

return 0;

}

if(bank() < change){

System.out.println("Здача неможлива. Сума монет менша за решту.\n");

randD = 0;

finish();

return 0;

}

int i = 0;

int[] virtNum = virtualNum(coins);

i = 0;

while(i < coins[0].length)

{

if (virtNum[i] != 0 && change - coins[0][i] >= 0){

change -= coins[0][i];

giving[gNum++] = i;

if (gNum == 1 || giving[uPos] != giving[gNum-1]) {

uPos = gNum - 1;

uNum++;

}

virtNum[i]--;

continue;

}

if (change == 0) break;

i++;

}

del(); //DELAY

if (change == 0) {

giveCoins(giving, gNum);

System.out.println("Здача видана:");

System.out.print(" ");

for(i = 0; i < coins[0].length; i++)

{

if (num(coins[0][i], giving) != 0){

if (num(coins[0][i], giving) > 1)

System.out.print(num(coins[0][i], giving) + "\*");

System.out.print(coins[0][i]);

if (--uNum != 0)System.out.print(" + ");

}

}

System.out.print(" = " + result + " коп.\n");

finish();

return 1;

}else

{

System.out.println(" Здача не можлива. Не вистачає монет.");

finish();

return 1;

}

}

}