1. Завдання до лабораторних занять 4
2. «Управління потоками»
3. Реалізуйте програмний код, даний у прикладі, та протестуйте її.
4. Модифікуйте програму, використовуючи методи управління потоками, так, щоб її робота була завжди корректно. Опрацюйте усі методи управління потоками на розробленій програмі та зробіть висновки.
5. Запропонуйте свій приклад роботи кількох потоків зі спільним об’єктом та опрацюйте на ньому конфлікт доступу до об’єкту.

/\*\*

author Cay Horstmann

\*/

public class UnsynchBankTest {

public static final int NACCOUNTS = 10;

public static final int INITIAL\_BALANCE = 10000;

public static void main(String[] args) {

Bank b = new Bank(NACCOUNTS, INITIAL\_BALANCE);

int i;

for (i = 0; i < NACCOUNTS; i++){

TransferThread t = new TransferThread(b, i,

INITIAL\_BALANCE);

t.setPriority(Thread.NORM\_PRIORITY + i % 2)

t.start () ;

 }

 }

 }

 class Bank {

public static final int NTEST = 10000;

private final int[] accounts;

private long ntransacts = 0;

/\*\*

@param n the number of accounts

@param initialBalance the initial balance

for each account

\*/

public Bank(int n, int initialBalance){

accounts = new int[n];

int i;

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

accounts[i] = initialBalance;

ntransacts = 0;

}

/\*\*

Sparam from the account to transfer from

cparam to the account to transfer to

@param amount the amount to transfer

\*/

public void transfer(int from, int to, int amount)

 throws InterruptedException{

accounts[from] -= amount;

accounts[to] += amount;

ntransacts++;

if (ntransacts % NTEST == 0)

test();

}

public void test(){

int sum = 0;

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

sum += accounts[i] ;

System.out.println("Transactions:" + ntransacts

+ " Sum: " + sum);

}

/\*\*

@return the number of accounts

\*/

public int size(){

return accounts.length;

}

 }

 class TransferThread extends Thread {

private Bank bank;

private int fromAccount;

private int maxAmount;

private static final int REPS = 1000;

**/\*\***

@param b the bank between whose account money is transferred

@param from the account to transfer money from

@param max the maximum amount of money in each transfer

\*/

public TransferThread(Bank b, int from, int max){

bank = b;

fromAccount = from;

maxAmount = max;

}

public void run(){

try{

while (!interrupted()){

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

int toAccount = (int)(bank.size() \* Math.random());

int amount = (int)(maxAmount \* Math.random() / REPS);

bank.transfer(fromAccount, toAccount, amount);

sleep(l);

}

}

} catch(InterruptedException e) {}

}

}