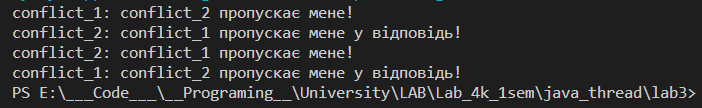


В першому варіанті я запустив код який був в методичці, вище наведений результат. З нього можна зрозуміти що 2 потоки конфліктують між собою і програма зависає.

Вирішенням задачі є використання захищеного призупинення. Ось результат:



import Myconflict.ConflictModify;

public class Main2 {

    public static void main(String[] args) throws InterruptedException {

        ConflictModify conflict\_1 = new ConflictModify("conflict\_1");

        ConflictModify conflict\_2 = new ConflictModify("conflict\_2");

        new MyThread2(conflict\_1, conflict\_2).start();

        new MyThread2(conflict\_2, conflict\_1).start();

    }

}

class MyThread2 extends Thread {

    ConflictModify conflict\_A = null;

    ConflictModify conflict\_B = null;

    public MyThread2(ConflictModify param, ConflictModify param2) {

        this.conflict\_A = param;

        this.conflict\_B = param2;

    }

    @Override

    public void run() {

        conflict\_B.checkIsInUse();

        conflict\_A.changeIsInUse(true);

        conflict\_B.changeIsInUse(true);

        conflict\_A.bow(conflict\_B);

        conflict\_A.changeIsInUse(false);

        conflict\_B.changeIsInUse(false);

    }

}

package Myconflict;

public class ConflictModify {

    private final String name;

    private boolean isInUse = false;

    public ConflictModify(String name) {

        this.name = name;

    }

    public String getName() {

        return this.name;

    }

    public synchronized void bow(ConflictModify bower) {

        System.out.format(

                "%s: %s" + " пропускає мене!%n",

                this.name,

                bower.getName());

        bower.bowBack(this);

    }

    public synchronized void bowBack(ConflictModify bower) {

        System.out.format(

                "%s: %s" + " пропускає мене у вiдповiдь!%n",

                this.name,

                bower.getName());

    }

    public synchronized void checkIsInUse() {

        while (isInUse) {

            try {

                wait();

            } catch (InterruptedException e) {}

        }

    }

    public synchronized void changeIsInUse(boolean var) {

        isInUse = var;

        notifyAll();

    }

}