Министерство образования и науки, молодежи и спорта Украины
ГВУЗ "Донецкий национальный технический университет"
кафедра Прикладной математики и информатики

Лабораторная работа №7

по курсу " Введение в программирование на Java"

по теме " Легковесные процессы и синхронизация"

Выполнил студент гр. ІПЗ-12а Егоров А. А.
Проверил: Середа А.А.

Донецк – 2014

Задание

1. Написать программу, которая создает минимум два подпроцесса (допускается больше двух), один из которых — управляющий, второй — вычислительный. Подпроцессы должны иметь доступ к общим разделяемым данным. Вычислительный подпроцесс выполняет вычисления по номеру варианта над разделяемыми данными. Управляющий подпроцесс передает данные вычислительному подпроцессу, выводит результат вычислений, а также может приостановить работу вычислительного подпроцесса.
2. Подпроцессы должны уведомлять друг друга о готовности очередной порции данных с помощью wait() и notify(). Синхронизировать подпроцессы таким образом, чтобы тесты, которые проверяют была ли выполнена такая синхронизация, считались не пройденными при отсутствии вызова wait().
3. Синхронизировать доступ к общим данным таким образом, чтобы тесты, которые проверяют была ли выполнена такая синхронизация, считались не пройденными при отсутствии ключевого слова synchronized.
4. Все классы описать внутри отдельного пакета.

Проверка года на високосность

Исходный код

Main.java

```
package lab.yegorov;

import lab.control.Handle;
/**
 * Created by AdminPC on 20.02.14.
 */
public class Main {
    public static void main(String args[]) {

        Thread handle = new Thread(new Handle());
        handle.start();
        try {
            handle.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Task.java

```
package lab.computability;

/**
 * Created by AdminPC on 27.02.14.
 */
public class Task implements Runnable {
    private LeapYear year;

    public Task(LeapYear year) {
        this.year = year;
    }
    @Override
    public void run() {
        year.Verify();
    }
}
```

Handle.java

```
package lab.control;

import lab.computability.LeapYear;
import lab.computability.Task;

import java.util.Scanner;

/**
 * Created by AdminPC on 20.02.14.
 */
public class Handle implements Runnable {
    private LeapYear year1;
    private LeapYear year2;
    private int y;

    public int inputData() {
        Scanner scan = new Scanner(System.in);
        int temp;
        while(true)
        try {
            System.out.print("Enter year\n>>> ");
            temp =  scan.nextInt();
            break;
        } catch (Exception e) {
            System.out.println("Exception. Try Again...");
            scan.nextLine();
        }
        return temp;
    }

    public void displayData(int y, boolean isLeap) {
        if(isLeap)
            System.out.println("" + y + " - leap year");
        else
            System.out.println("" + y + " - no leap year");
    }
```

```
    @Override
    public void run() {
        y = inputData();
        year1 = new LeapYear(y);
        y = inputData();
        year2 = new LeapYear(y);


        Thread compute1 = new Thread(new Task(year1));
        Thread compute2 = new Thread(new Task(year2));
        compute1.start();
        compute2.start();


        displayData(year1.getYear(), year1.getLeapYear());
        displayData(year2.getYear(), year2.getLeapYear());
     }
}
```

LeapYear.java

```
package lab.computability;
/**
 * Created by AdminPC on 20.02.14.
 */
public class LeapYear  {
    private int year;
    private boolean isLeapYear;
    private boolean isYearCompute;

    public LeapYear(int year) {
        this.year = year;
        isYearCompute = false;
    }

    public synchronized void Verify() {
        while(isYearCompute)
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
```

```
            isLeapYear =  (year % 4 == 0) && (year % 100 != 0) || (year % 400 == 0);
            isYearCompute = true;
            notify();
        }

        public synchronized boolean getLeapYear() {
            while(!isYearCompute) {
                try {
                    wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            isYearCompute = false;
            notify();
            return isLeapYear;
        }

        public int getYear() {
            return year;
        }
    }
```

Экранные формы