Министерство Образования и Науки Украины

Донецкий Национальный Технический Университет

Кафедра: Прикладной Математики и Информатики

Лабораторная работа № 6

По предмету «Введение в программирование .NET и Java»

Тема**:** «Легковесные процессы и синхронизация»

Подготовил:

Лысенко А.С.

Группа: ПИ-13б

Проверил:

асс. каф. ПМИ Павловский Евгений Викторович

Красноармейск 2015

**Задание на лабораторную работу:**

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.

|  |  |
| --- | --- |
| 7 | По заданной длине ребра куба найти площадь грани куба |

Код на Java

**package** ThreadPkg;

**import** java.util.\*;

**class** Count

{

**int** d;

**boolean** f = **false**;

**synchronized** **public** **void** cube()

{

**if**(!f)

{

**try**

{

Scanner scn = **new** Scanner(System.***in***);

System.***out***.print("Введите кол-во кубов : ");

d = scn.nextInt();

**double**[] n = **new** **double** [d];

System.***out***.println("Получено : " + d);

spacecube(d);

wait();

}

**catch** (InterruptedException e)

{

System.***out***.println("InterruptedException выброшено");

}

}

f = **false**;

notify();

}

**synchronized** **public** **void** spacecube(**int** d)

{

**if**(!f)

{

**try**

{

Scanner scn = **new** Scanner(System.***in***);

**double**[] n = **new** **double** [d];

**int** i;

**double** x = 0;

**double** p = 0;

**for**(i = 0; i < d; i++)

{

System.***out***.print("Введите длину куба[" + (i + 1) + "] : ");

n[i] = scn.nextInt();

p = n[i];

x = Math.*pow*(p, 2);

n[i] = 6 \* x;

}

System.***out***.println("Отдано : " + d);

**for**(i = 0; i < d; i++)

{

System.***out***.println("Площадь вашего куба[" + (i + 1) + "] : " + n[i]);

}

wait();

}

**catch** (InterruptedException e)

{

System.***out***.println("InterruptedException выброшено");

}

}

**this**.d = d;

f = **true**;

notify();

}

}

**class** Cube **implements** Runnable

{

Count z;

Thread t;

Cube(Count z)

{

**this**.z = z;

t = **new** Thread(**this**, "Cube");

t.start();

}

**public** **void** run()

{

**while**(**true**)

{

z.cube();

}

}

}

**class** EnterCube **implements** Runnable

{

Count z;

Thread t;

EnterCube(Count z)

{

**this**.z = z;

t = **new** Thread(**this**, "EnterCube");

t.start();

}

**public** **void** run()

{

**int** d = 0;

**while**(**true**)

{

z.spacecube(d);

}

}

}

**public** **class** MyThreadTest

{

**public** **static** **void** main(String args[])

{

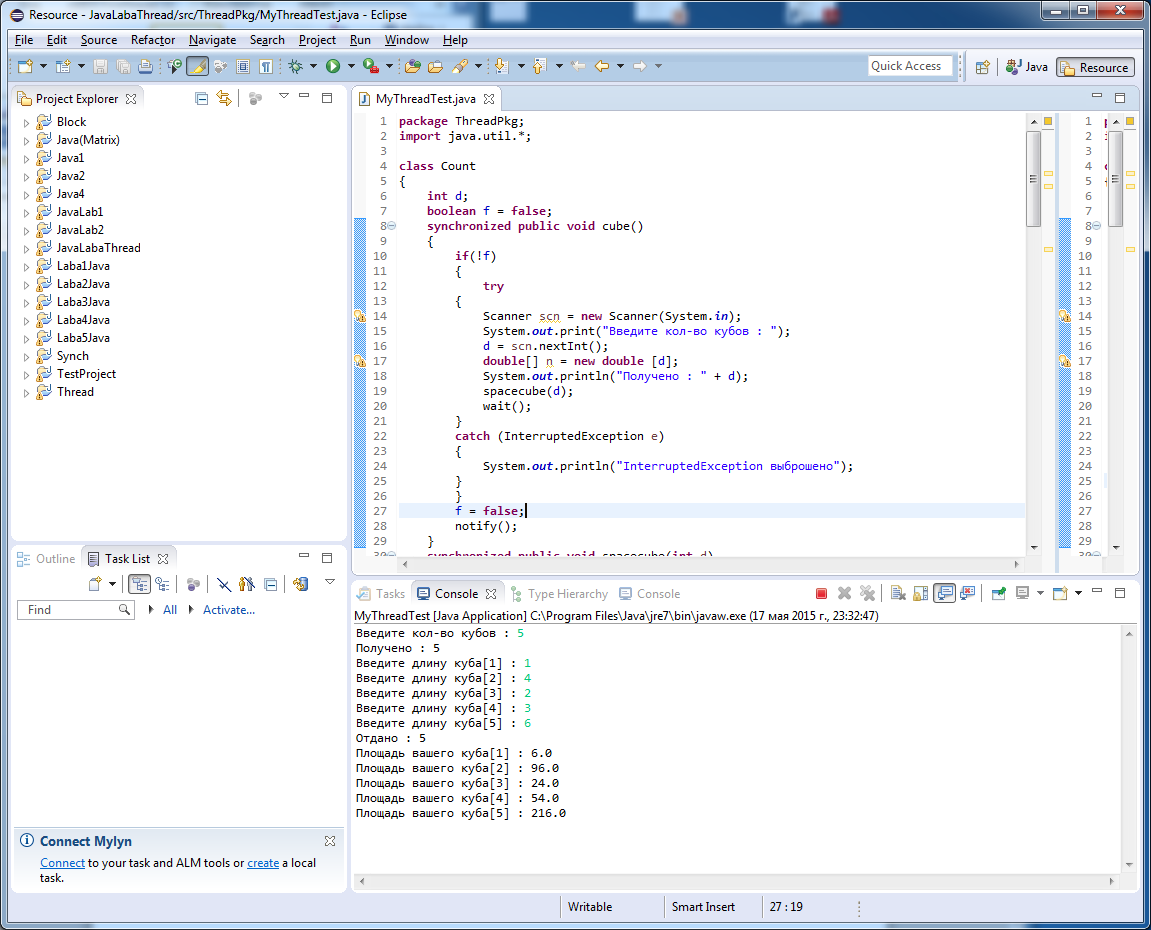
//System.out.println("Для прерывания нажмите Control - C.");

Count z = **new** Count();

**new** Cube(z);

}

}



Введите кол-во кубов : 5

Получено : 5

Введите длину куба[1] : 1

Введите длину куба[2] : 4

Введите длину куба[3] : 2

Введите длину куба[4] : 3

Введите длину куба[5] : 6

Отдано : 5

Площадь вашего куба[1] : 6.0

Площадь вашего куба[2] : 96.0

Площадь вашего куба[3] : 24.0

Площадь вашего куба[4] : 54.0

Площадь вашего куба[5] : 216.0