**package** Lab6;

**import** java.util.Calendar;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Laba6. Java. Monitors

\*\* **@author** Matviychyk B

\*\* IO - 91 group

\*\* A=B\*(MO\*MX)\*l+Z(MR\*MT)

\*\* OR: MX,B,l,Z,MT

\*\* 08,05,2012

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **class** Laba6 {

**static** **int** *N* = 12;

**static** **int** *P* = 6;

**static** **int** *H* = *N*/*P*;

**static** **int**[] *A* = **new** **int**[*N*];

**static** **int**[] *Bo* = **new** **int**[*N*];

**static** **int**[] *Zo* = **new** **int**[*N*];

**static** **int**[][] *MO* = **new** **int**[*N*][*N*];

**static** **int**[][] *MR* = **new** **int**[*N*][*N*];

**static** **int**[][] *MXo* = **new** **int**[*N*][*N*];

**static** **int**[][] *MTo* = **new** **int**[*N*][*N*];

**static** **int** *lo*;

**static** Work *w* = **new** Work();

**static** Monitor *m* = **new** Monitor();

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

**long** start,stop;

Calendar cal = Calendar.*getInstance*();

start = cal.getTimeInMillis();

System.*out*.println("Laba6 started");

T1 f1 = **new** T1("task1");

T2 f2 = **new** T2("task2");

T3 f3 = **new** T3("task3");

T4 f4 = **new** T4("task4");

T5 f5 = **new** T5("task5");

T6 f6 = **new** T6("task6");

f1.start();

f2.start();

f3.start();

f4.start();

f5.start();

f6.start();

**try** {

f2.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**try** {

f3.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**try** {

f4.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**try** {

f5.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**try** {

f6.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**try** {

f1.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

cal = Calendar.*getInstance*();

stop = cal.getTimeInMillis();

**double** t= stop-start;

**if** (t>=100){

t=t\*0.1;

}

System.*out*.println("Laba6 finished."+'\n');

}// main

}// Runner

**package** Lab6;

**public** **class** Monitor {

**private** **static** **int** *F1* = 0;

**private** **static** **int** *F2* = 0;

**private** **static** **int** *F3* = 0;

**private** **static** **int** *F4* = 0;

**private** **static** **int** *F5* = 0;

**private** **static** **int** *N* = Laba6.*N*;

**private** **int** l = 0;

**private** **int**[] B = **new** **int**[*N*];

**private** **int**[] Z = **new** **int**[*N*];

**private** **int**[][] MX = **new** **int**[*N*][*N*];

**private** **int**[][] MT = **new** **int**[*N*][*N*];

//input

**public** **synchronized** **void** setl(**int** l) {

**this**.l = l;

}

**public** **synchronized** **void** setB(**int**[] B) {

**this**.B = B;

}

**public** **synchronized** **void** setZ(**int**[] Z) {

**this**.Z = Z;

}

**public** **synchronized** **void** setMX(**int**[][] MX) {

**this**.MX = MX;

}

**public** **synchronized** **void** setMT(**int**[][] MT) {

**this**.MT = MT;

}

//copy

**public** **int** copyl() {

**return** l;

}

**public** **int**[] copyB() {

**return** B;

}

**public** **int**[] copyZ() {

**return** Z;

}

**public** **int**[][] copyMX() {

**return** MX;

}

**public** **int**[][] copyMT() {

**return** MT;

}

//signal

**public** **synchronized** **void** SignalInput() {

*F1*++;

**if**(*F1* == 3) notifyAll();

}

**public** **synchronized** **void** SignalCalc() {

*F2*++;

**if**(*F2* == 6) notifyAll();

}

//entry

**public** **synchronized** **void** WaitInput() {

**try**{

**while**(*F1* != 3) wait();

} **catch**(Exception e) {}

}

**public** **synchronized** **void** WaitCalc() {

**try**{

**while**(*F2* != 6) wait();

} **catch**(Exception e) {}

}

}//Monitor

**package** Lab6;

**public** **class** Work {

**int** n = Laba6.*N*;

/\*\*

\* Ввод вектора.

\* **@return** вектор.

\*/

**public** **int**[] VectorInput() {

**int**[] vector = **new** **int**[n];

**for** (**int** i = 0; i < n; i++) {

vector[i] = 1;

}

**return** vector;

}

/\*\*

\* Ввод матрицы.

\* **@return** матрицу.

\*/

**public** **int**[][] MatrixInput() {

**int**[][] matrix = **new** **int**[n][n];

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0; j < n; j++) {

matrix[i][j] = 1;

}

}

**return** matrix;

}

/\*\*

\* Вывод вектора.

\* **@param** vector - введенный вектор;

\*/

**public** **void** VectorOutput(**int**[] vector) {

**if** (n <= 18) {

**for** (**int** i = 0; i < n; i++) {

System.*out*.print(vector[i] + " ");

}

System.*out*.println();

}

}

/\*\*

\* Вывод матрицы.

\* **@param** matrix - введенная матрица;

\*/

**public** **void** MatrixOutput(**int**[][] matrix) {

**if** (n <= 18) {

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0; j < n; j++) {

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

}

System.*out*.println();

}

System.*out*.println();

}

}

/\*\*

\* Умножение числа на вектор.

\* **@param** One - левая граница поиска;

\* **@param** Two - правая граница поиска;

\* **@param** Q - текущий максимум;

\* **@param** vector - умножаемый вектор;

\* **@param** vectorR - результирующий вектор;

\*/

**public** **void** NumbVectorMnozh(**int** One, **int** Two, **int** Q, **int**[] vector, **int**[] vectorR) {

**for** (**int** i = One; i < Two; i++) {

vectorR[i] = Q \* vector[i];

}

}

/\*\*

\* Умножение матриц.

\* **@param** One - левая граница поиска;

\* **@param** Two - правая граница поиска;

\* **@param** matrix1 - матрица-множитель;

\* **@param** matrix2 - множимая матрица;

\* **@param** matrixR - результирующая матрица;

\*/

**public** **void** MatrixMnozh(**int** One, **int** Two, **int**[][] matrix1, **int**[][] matrix2, **int**[][] matrixR) {

**for** (**int** i = One; i < Two; i++) {

**for** (**int** j = 0; j < matrix1.length; j++) {

matrixR[i][j] = 0;

**for** (**int** k = 0; k < matrixR.length; k++) {

matrixR[i][j] += matrix1[k][j] \* matrix2[i][k];

}

}

}

}

/\*\*

\* Умножение вектора на матрицу.

\* **@param** One - левая граница поиска;

\* **@param** Two - правая граница поиска;

\* **@param** vector - вектор-множитель;

\* **@param** matrix - множимая матрица;

\* **@param** vectorR - результирующий вектор;

\*/

**public** **void** VectorMatrixMnozh(**int** One, **int** Two, **int**[] vector, **int**[][] matrix, **int**[] vectorR) {

**for** (**int** i = One; i < Two; i++) {

vectorR[i] = 0;

**for** (**int** j = 0; j < vectorR.length; j++) {

vectorR[i] += vector[j] \* matrix[i][j];

}

}

}

/\*\*

\* сложение векторов.

\* **@param** One - левая граница поиска;

\* **@param** Two - правая граница поиска;

\* **@param** vector1 - вектор-множитель;

\* **@param** vector2 - множимый вектор;

\* **@param** vectorR - результирующий вектор;

\*/

**public** **void** VecSlozh(**int** One, **int** Two, **int**[] vector1, **int**[] vector2, **int**[] vectorR) {

**for** (**int** i = One; i < Two; i++) {

vectorR[i] = vector1[i] + vector2[i];

}

}

}//Work

**package** Lab6;

**public** **class** T1 **extends** Thread {

**int** n = Laba6.*N*;

**int** H = Laba6.*H*;

**int** One = 0\*H;

**int** Two = 1\*H;

**int** l1;

**int**[] B1;

**int**[] Z1;

**int**[][] MX1;

**int**[][] MT1;

**int** [] S1= **new** **int** [n];

**int** [] S2= **new** **int** [n];

**int** [] S3= **new** **int** [n];

**int** [][]MB1 = **new** **int** [n][n];

**int** [][]MB2 = **new** **int** [n][n];

**public** T1(String s) {

**this**.setName(s);

}

/\*\*

\* Действия при выполнении T1.

\*/

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

System.*out*.println(**this**.getName() + " started");

Laba6.*MTo* = Laba6.*w*.MatrixInput();

Laba6.*MXo* = Laba6.*w*.MatrixInput();

Laba6.*m*.setMT(Laba6.*MTo*);

Laba6.*m*.setMX(Laba6.*MXo*);

Laba6.*m*.SignalInput();

Laba6.*m*.WaitInput();

l1 = Laba6.*m*.copyl();

B1 = Laba6.*m*.copyB();

Z1 = Laba6.*m*.copyZ();

MX1 = Laba6.*m*.copyMX();

MT1 = Laba6.*m*.copyMT();

Laba6.*w*.MatrixMnozh(One, Two, MX1, Laba6.*MO*, MB1);

Laba6.*w*.VectorMatrixMnozh(One, Two, B1, MB1, S1);

Laba6.*w*.NumbVectorMnozh(One, Two, l1, S1, S2);

Laba6.*w*.MatrixMnozh(One, Two, MT1, Laba6.*MR*, MB2);

Laba6.*w*.VectorMatrixMnozh(One, Two, Z1, MB2, S3);

Laba6.*w*.VecSlozh(One, Two, S2, S3, Laba6.*A*);

Laba6.*m*.SignalCalc();

Laba6.*m*.WaitCalc();

System.*out*.println(**this**.getName() + " finished.");

System.*out*.println();

}//run

}//T1

**package** Lab6;

**public** **class** T2 **extends** Thread {

**int** n = Laba6.*N*;

**int** H = Laba6.*H*;

**int** One = 1\*H;

**int** Two = 2\*H;

**int** l2;

**int**[] B2;

**int**[] Z2;

**int**[][] MX2;

**int**[][] MT2;

**int** [] S1= **new** **int** [n];

**int** [] S2= **new** **int** [n];

**int** [] S3= **new** **int** [n];

**int** [][]MB1= **new** **int** [n][n];

**int** [][]MB2= **new** **int** [n][n];

**public** T2(String s) {

**this**.setName(s);

}

/\*\*

\* Действия при выполнении T1.

\*/

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

System.*out*.println(**this**.getName() + " started");

Laba6.*MO* = Laba6.*w*.MatrixInput();

Laba6.*MR* = Laba6.*w*.MatrixInput();

Laba6.*m*.SignalInput();

Laba6.*m*.WaitInput();

l2 = Laba6.*m*.copyl();

B2 = Laba6.*m*.copyB();

Z2 = Laba6.*m*.copyZ();

MX2 = Laba6.*m*.copyMX();

MT2 = Laba6.*m*.copyMT();

Laba6.*w*.MatrixMnozh(One, Two, MX2, Laba6.*MO*, MB1);

Laba6.*w*.VectorMatrixMnozh(One, Two, B2, MB1, S1);

Laba6.*w*.NumbVectorMnozh(One, Two, l2, S1, S2);

Laba6.*w*.MatrixMnozh(One, Two, MT2, Laba6.*MR*, MB2);

Laba6.*w*.VectorMatrixMnozh(One, Two, Z2, MB2, S3);

Laba6.*w*.VecSlozh(One, Two, S2, S3, Laba6.*A*);

Laba6.*m*.SignalCalc();

Laba6.*m*.WaitCalc();

System.*out*.println(**this**.getName() + " finished.");

System.*out*.println();

}//run

}

**package** Lab6;

**public** **class** T3 **extends** Thread {

**int** n = Laba6.*N*;

**int** H = Laba6.*H*;

**int** One = 2\*H;

**int** Two = 3\*H;

**int** l3;

**int**[] B3;

**int**[] Z3;

**int**[][] MX3;

**int**[][] MT3;

**int** [] S1= **new** **int** [n];

**int** [] S2= **new** **int** [n];

**int** [] S3= **new** **int** [n];

**int** [][]MB1= **new** **int** [n][n];

**int** [][]MB2= **new** **int** [n][n];

**public** T3(String s) {

**this**.setName(s);

}

/\*\*

\* Действия при выполнении T1.

\*/

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

System.*out*.println(**this**.getName() + " started");

Laba6.*m*.WaitInput();

l3 = Laba6.*m*.copyl();

B3 = Laba6.*m*.copyB();

Z3 = Laba6.*m*.copyZ();

MX3 = Laba6.*m*.copyMX();

MT3 = Laba6.*m*.copyMT();

Laba6.*w*.MatrixMnozh(One, Two, MX3, Laba6.*MO*, MB1);

Laba6.*w*.VectorMatrixMnozh(One, Two, B3, MB1, S1);

Laba6.*w*.NumbVectorMnozh(One, Two, l3, S1, S2);

Laba6.*w*.MatrixMnozh(One, Two, MT3, Laba6.*MR*, MB2);

Laba6.*w*.VectorMatrixMnozh(One, Two, Z3, MB2, S3);

Laba6.*w*.VecSlozh(One, Two, S2, S3, Laba6.*A*);

Laba6.*m*.SignalCalc();

Laba6.*m*.WaitCalc();

System.*out*.println(**this**.getName() + " finished.");

System.*out*.println();

}//run

}

**package** Lab6;

**public** **class** T4 **extends** Thread {

**int** n = Laba6.*N*;

**int** H = Laba6.*H*;

**int** One = 3\*H;

**int** Two = 4\*H;

**int** l4;

**int**[] B4;

**int**[] Z4;

**int**[][] MX4;

**int**[][] MT4;

**int** [] S1= **new** **int** [n];

**int** [] S2= **new** **int** [n];

**int** [] S3= **new** **int** [n];

**int** [][]MB1= **new** **int** [n][n];

**int** [][]MB2= **new** **int** [n][n];

**public** T4(String s) {

**this**.setName(s);

}

/\*\*

\* Действия при выполнении T1.

\*/

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

System.*out*.println(**this**.getName() + " started");

Laba6.*m*.WaitInput();

l4 = Laba6.*m*.copyl();

B4 = Laba6.*m*.copyB();

Z4 = Laba6.*m*.copyZ();

MX4 = Laba6.*m*.copyMX();

MT4 = Laba6.*m*.copyMT();

Laba6.*w*.MatrixMnozh(One, Two, MX4, Laba6.*MO*, MB1);

Laba6.*w*.VectorMatrixMnozh(One, Two, B4, MB1, S1);

Laba6.*w*.NumbVectorMnozh(One, Two, l4, S1, S2);

Laba6.*w*.MatrixMnozh(One, Two, MT4, Laba6.*MR*, MB2);

Laba6.*w*.VectorMatrixMnozh(One, Two, Z4, MB2, S3);

Laba6.*w*.VecSlozh(One, Two, S2, S3, Laba6.*A*);

Laba6.*m*.SignalCalc();

Laba6.*m*.WaitCalc();

System.*out*.println(**this**.getName() + " finished.");

System.*out*.println();

}//run

}

**package** Lab6;

**public** **class** T5 **extends** Thread {

**int** n = Laba6.*N*;

**int** H = Laba6.*H*;

**int** One = 4\*H;

**int** Two = 5\*H;

**int** l5;

**int**[] B5;

**int**[] Z5;

**int**[][] MX5;

**int**[][] MT5;

**int** [] S1= **new** **int** [n];

**int** [] S2= **new** **int** [n];

**int** [] S3= **new** **int** [n];

**int** [][]MB1= **new** **int** [n][n];

**int** [][]MB2= **new** **int** [n][n];

**public** T5(String s) {

**this**.setName(s);

}

/\*\*

\* Действия при выполнении T1.

\*/

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

System.*out*.println(**this**.getName() + " started");

Laba6.*m*.WaitInput();

l5 = Laba6.*m*.copyl();

B5 = Laba6.*m*.copyB();

Z5 = Laba6.*m*.copyZ();

MX5 = Laba6.*m*.copyMX();

MT5 = Laba6.*m*.copyMT();

Laba6.*w*.MatrixMnozh(One, Two, MX5, Laba6.*MO*, MB1);

Laba6.*w*.VectorMatrixMnozh(One, Two, B5, MB1, S1);

Laba6.*w*.NumbVectorMnozh(One, Two, l5, S1, S2);

Laba6.*w*.MatrixMnozh(One, Two, MT5, Laba6.*MR*, MB2);

Laba6.*w*.VectorMatrixMnozh(One, Two, Z5, MB2, S3);

Laba6.*w*.VecSlozh(One, Two, S2, S3, Laba6.*A*);

Laba6.*m*.SignalCalc();

Laba6.*m*.WaitCalc();

System.*out*.println(**this**.getName() + " finished.");

System.*out*.println();

}//run

}

**package** Lab6;

**public** **class** T6 **extends** Thread {

**int** n = Laba6.*N*;

**int** H = Laba6.*H*;

**int** One = 5\*H;

**int** Two = 6\*H;

**int** l6;

**int**[] B6;

**int**[] Z6;

**int**[][] MX6;

**int**[][] MT6;

**int** [] S1= **new** **int** [n];

**int** [] S2= **new** **int** [n];

**int** [] S3= **new** **int** [n];

**int** [][]MB1= **new** **int** [n][n];

**int** [][]MB2= **new** **int** [n][n];

**public** T6(String s) {

**this**.setName(s);

}

/\*\*

\* Действия при выполнении T1.

\*/

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

System.*out*.println(**this**.getName() + " started");

Laba6.*lo* = 1;

Laba6.*Bo* = Laba6.*w*.VectorInput();

Laba6.*Zo* = Laba6.*w*.VectorInput();

Laba6.*m*.setB(Laba6.*Bo*);

Laba6.*m*.setZ(Laba6.*Zo*);

Laba6.*m*.setl(Laba6.*lo*);

Laba6.*m*.SignalInput();

Laba6.*m*.WaitInput();

l6 = Laba6.*m*.copyl();

B6 = Laba6.*m*.copyB();

Z6 = Laba6.*m*.copyZ();

MX6 = Laba6.*m*.copyMX();

MT6 = Laba6.*m*.copyMT();

Laba6.*w*.MatrixMnozh(One, Two, MX6, Laba6.*MO*, MB1);

Laba6.*w*.VectorMatrixMnozh(One, Two, B6, MB1, S1);

Laba6.*w*.NumbVectorMnozh(One, Two, l6, S1, S2);

Laba6.*w*.MatrixMnozh(One, Two, MT6, Laba6.*MR*, MB2);

Laba6.*w*.VectorMatrixMnozh(One, Two, Z6, MB2, S3);

Laba6.*w*.VecSlozh(One, Two, S2, S3, Laba6.*A*);

Laba6.*m*.SignalCalc();

Laba6.*m*.WaitCalc();

Laba6.*w*.VectorOutput(Laba6.*A*);

System.*out*.println(**this**.getName() + " finished.");

System.*out*.println();

}//run

}//T1