Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Лабораторна робота №3

з дисципліни «Паралельне програмування -1.

Основи паралельного програмування»

Тема: **Потоки в мові Java**

Виконав

студент групи ІП-31

Кобилинський Дмитро

Київ 2015

**Цель работы:** изучение средств языка Java для работы с потоками.

**Выполнение работы:** Разработать программу, содержащую п а р а л л е л ь н ы е потоки,

каждый из которых реализует функцию F1, F2, F3 из лабораторной работы номер 1.

Требования к созданию потоков и необходимые исследования программы описаны в

лабораторной работе 2.

В потоках использовать методы sleep() и join().

|  |
| --- |
| Lab\_3.java |

1 */\*\**

2  *\* Created by Dmytriy Kobylynskiy on 02.10.2015.*

3  *\* Parallel programming*

4  *\* IP-31*

5  *\* F1 = MIN(A + B) \* (B + C) \*(MA\*MD)*

6  *\* F2 = ML + MK \*MO*

7  *\* F3 = SORT(S + T)\*TRANS(MS\*MR)*

8  *\*/*

9 **public class** Lab\_3 {

10

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

12

13 Thread t1 = **new** Thread(**null**, **new** FunctionOne(), **"Task 1"**, 1000);

14 Thread t2 = **new** Thread(**null**, **new** FunctionTwo(), **"Task 2"**, 1000);

15 Thread t3 = **new** Thread(**null**, **new** FunctionThree(), **"Task 3"**, 1000);

16 t1.setPriority(1);

17 t2.setPriority(4);

18 t3.setPriority(5);

19 t1.start();

20 t2.start();

21 t2.join();

22 t3.start();

23

24 }

25 }

26

|  |
| --- |
| Data.java |

1 */\*\**

2  *\* Created by Dmytriy Kobylynskiy on 03.10.2015.*

3  *\*/*

4 **public class** Data {

5 **int** size ;

6

7 **public** Data(**int** n){

8 **this**.size = n;

9 }

10

11 */\*\**

12  *\* Method for output matrix*

13  *\** ***@param*** *matrix*

14  *\*/*

15 **public void** outputMatrix(**int**[][] matrix){

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

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

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

19 }

20 System.out.println();

21 }

22 }

23

24 */\*\**

25  *\* Method for output vector*

26  *\** ***@param*** *vector*

27  *\*/*

28 **public void** outputVector(**int**[] vector){

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

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

31 }

32 }

33

34 */\*\**

35  *\* Function for input matrix*

36  *\** ***@return*** *matrix*

37  *\*/*

38 **public int**[][] inputMatrix(){

39 **int**[][] matrix = **new int**[size][size];

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

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

42 matrix[i][j] = 1;

43 }

44 }

45 **return** matrix;

46 }

47

48 */\*\**

49  *\* Function for input vector*

50  *\** ***@return*** *vector*

51  *\*/*

52 **public int**[] inputVector(){

53 **int**[] vector = **new int**[size];

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

55 vector[i] = 1;

56 }

57 **return** vector;

58 }

59

60 */\*\**

61  *\* Function for addition*

62  *\** ***@param*** *add1 - addition matrix*

63  *\** ***@param*** *add2 - addition matrix*

64  *\** ***@return*** *result matrix*

65  *\*/*

66 **public int**[][] sumMatrix(**int**[][] add1,**int**[][] add2){

67 **int**[][] result = **new int**[size][size];

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

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

70 result[i][j] = add1[i][j] + add2[i][j];

71 }

72 }

73 **return** result;

74 }

75

76 */\*\**

77  *\* Function for multiply matrix on vector*

78  *\** ***@param*** *m1 - multiplier matrix*

79  *\** ***@param*** *v1 - multiplier vector*

80  *\** ***@return*** *result vector*

81  *\*/*

82 **public int**[] multiplyMatrixOnVector(**int**[][] m1, **int**[] v1){

83 **int**[] R = **new int**[size];

84

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

86 **int** s = 0;

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

88 s += m1[i][j] \* v1[j];

89 }

90 R[i] = s;

91 }

92 **return** R;

93 }

94

95 */\*\**

96  *\* Function for multiply matrix*

97  *\** ***@param*** *m1 - multiplier matrix*

98  *\** ***@param*** *m2 - multiplier matrix*

99  *\** ***@return*** *result matrix*

100  *\*/*

101

102 **public int**[][] multiplyMatrix(**int**[][] m1, **int**[][] m2){

103 **int**[][] result = **new int**[size][size];

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

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

106 **int** s = 0;

107 **for** (**int** k = 0; k < size; k++) {

108 s += m1[i][k] \* m2[k][j];

109 result[i][j] = s;

110 }

111 }

112 }

113 **return** result;

114 }

115

116 */\*\**

117  *\* Function for multiply matrix on digit*

118  *\** ***@param*** *m1 - multiplier matrix*

119  *\** ***@param*** *digit - digit*

120  *\** ***@return***

121  *\*/*

122 **public int**[][] multiplyMatrixOnDigit(**int**[][] m1,**int** digit){

123 **int**[][] result = **new int**[size][size];

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

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

126 result[i][j]=digit\*m1[i][j];

127 }

128 }

129 **return** result;

130 }

131

132 */\*\**

133  *\* Function for search min element in matrix*

134  *\** ***@param*** *m1 - matrix for search*

135  *\** ***@return*** *min element*

136  *\*/*

137 **public int** minMatrix(**int**[][] m1){

138 **int** result = m1[0][0];

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

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

141 **if**(result<m1[i][j]){

142 result=m1[i][j];

143 }

144 }

145 }

146 **return** result;

147 }

148

149 */\*\**

150  *\* Function for transpose matrix*

151  *\** ***@param*** *m1 - matrix*

152  *\** ***@return*** *- transpose matrix*

153  *\*/*

154 **public int**[][] transpose(**int**[][] m1){

155 **int**[][] result = **new int**[size][size];

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

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

158 result[i][j]=m1[j][i];

159 }

160 }

161 **return** result;

162 }

163

164 */\*\**

165  *\* Function for addition vector*

166  *\** ***@param*** *add1 - addition vector*

167  *\** ***@param*** *add2 - addition vector*

168  *\** ***@return*** *resutl vector*

169  *\*/*

170 **public int**[] sumVector(**int**[] add1,**int**[] add2){

171 **int**[] result = **new int**[size];

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

173 result[i] = add1[i] + add2[i];

174 }

175 **return** result;

176 }

177

178 */\*\**

179  *\* Function for multiply vector on digit*

180  *\** ***@param*** *v1 vector*

181  *\** ***@param*** *digit digit*

182  *\** ***@return*** *result vector*

183  *\*/*

184 **public int**[] multiplyVectorOnDigit(**int**[] v1,**int** digit){

185 **int**[] result = **new int**[size];

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

187 result[i]=digit\*v1[i];

188 }

189 **return** result;

190 }

191

192 */\*\**

193  *\* Function for search min element of vector*

194  *\** ***@param*** *v1 vector*

195  *\** ***@return*** *min element of vector*

196  *\*/*

197 **public int** minVector(**int**[] v1){

198 **int** result = v1[0];

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

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

201 **if**(result<v1[i]){

202 result=v1[i];

203 }

204 }

205 }

206 **return** result;

207 }

208

209 */\*\**

210  *\* Function for sorting vector*

211  *\** ***@param*** *v - vector*

212  *\** ***@return*** *sorted vector*

213  *\*/*

214 **public int**[] sortVector(**int**[] v){

215 **int** k;

216 **for**(**int** i=0;i<size-1;i++){

217 **for**(**int** j=0;j<size-i-1;j++){

218 **if**(v[j]>v[j+1]){

219 k=v[j];

220 v[j]=v[j+1];

221 v[j+1]=k;

222 }

223 }

224 }

225 **return** v;

226 }

227 }

|  |
| --- |
| FunctionOne.java |

1 */\*\**

2  *\* Created by Dmytriy Kobylynskiy on 02.10.2015.*

3  *\*/*

4 **public class** FunctionOne **extends** Data **implements** Runnable {

5

6 **public** FunctionOne(){

7 **super**(1000);

8 }

9

10 @Override

11 **public void** run(){

12 System.out.println(**"Task 1 started"**);

13 **int**[] A,B,C,D,E;

14 **int**[][] MA,MD,MB,MC;

15 A = inputVector();

16 B = inputVector();

17 C = inputVector();

18 MA = inputMatrix();

19 MD = inputMatrix();

20 D = multiplyVectorOnDigit(multiplyMatrixOnVector(multiplyMatrix(MA, MD), sumVector(B, C)), minVector(sumVector(A, B)));

21 System.out.println(**"Task 1 finished"**);

22 }

23 }

|  |
| --- |
| FunctionTwo.java |

1 */\*\**

2  *\* Created by Dmytriy Kobylynskiy on 02.10.2015.*

3  *\*/*

4 **public class** FunctionTwo **extends** Data **implements** Runnable {

5 **public** FunctionTwo(){

6 **super**(1000);

7 }

8

9 @Override

10 **public void** run() {

11 System.out.println(**"Task 2 started"**);

12 **int**[][] MM,ML,MK,MO;

13 ML = inputMatrix();

14 MK = inputMatrix();

15 MO = inputMatrix();

16 MM = sumMatrix(ML, multiplyMatrix(MK, MO));

17 System.out.println(**"Task 2 finished"**);

18 }

19 }

20

|  |
| --- |
| FunctionThree.java |

1 */\*\**

2  *\* Created by Dmytriy Kobylynskiy on 02.10.2015.*

3  *\*/*

4 **public class** FunctionThree **extends** Data **implements** Runnable {

5 **public** FunctionThree(){

6 **super**(1000);

7 }

8

9 @Override

10 **public void** run(){

11 System.out.println(**"Task 3 started"**);

12 **int**[] S,T,R;

13 **int**[][] MS,MR;

14 S = inputVector();

15 T = inputVector();

16 MS = inputMatrix();

17 MR = inputMatrix();

18 R = multiplyMatrixOnVector(transpose(multiplyMatrix(MS, MR)), sortVector(sumVector(S, T)));

19 System.out.println(**"Task 3 finished"**);

20 }

21 }

22