Міністерство освіти і науки, молоді та спорту України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Звіт**

**до лабораторної роботи №3**

**Виконав:**

Ст. гр. КН-203 Павлишин  Д. А.

**Перевірив:**

Скрибайло-Леськів Д.Ю.

**Завдання**

Exercise 1: (1) Write a program that prints values from 1 to 100.

Exercise 2: (2) Write a program that generates 25 random int values. For each value, use an

if-else statement to classify it as greater than, less than, or equal to a second randomly generated

value.

Exercise 3: (1) Modify Exercise 2 so that your code is surrounded by an “infinite” while loop.

It will then run until you interrupt it from the keyboard (typically by pressing ControlC).

Exercise 4: (3) Write a program that uses two nested for loops and the modulus operator (%)

to detect and print prime numbers (integral numbers that are not evenly divisible by any other

numbers except for themselves and 1).

Exercise 5: (4) Repeat Exercise 10 from the previous chapter, using the ternary operator and a

bitwise test to display the ones and zeroes, instead of Integer.toBinaryString( ).

Exercise 6: (2) Modify the two test( ) methods in the previous two programs so that they take

two extra arguments, begin and end, and so that testval is tested to see if it is within the range

between (and including) begin and end.

Exercise 6: (2) Modify the two test( ) methods in the previous two programs so that they take

two extra arguments, begin and end, and so that testval is tested to see if it is within the range

between (and including) begin and end.

Exercise 7: (1) Modify Exercise 1 so that the program exits by using the break keyword at

value 99. Try using return instead.

Exercise 8: (2) Create a switch statement that prints a message for each case, and put the

switch inside a for loop that tries each case. Put a break after each case and test it, then

remove the breaks and see what happens.

Exercise 9: (4) A Fibonacci sequence is the sequence of numbers 1, 1, 2, 3, 5, 8, 13, 21, 34, and

so on, where each number (from the third on) is the sum of the previous two. Create a method

that takes an integer as an argument and displays that many Fibonacci numbers starting from the

beginning, e.g., If you run java Fibonacci 5 (where Fibonacci is the name of the class) the

output will be: 1, 1, 2, 3, 5.

**Код програми:**

**LabThree.java:**

package com.pavlyshyn;  
  
import java.util.Random;  
  
public class LabThree {  
  
 public static void main(String[] args) {  
 //Task 1  
 System.*out*.println("Task 1");  
 for (int i = 1; i <= 100; i++) {  
 System.*out*.println(i);  
 }  
  
 //Task 2  
 System.*out*.println("Task 2");  
 Random random = new Random();  
 int RandNumber1 = 0, RandNumber2 = 0;  
 for (int i = 0; i < 25; i++) {  
 int RandNumber3 = random.nextInt(100);  
 if (RandNumber3 > RandNumber2) {  
 RandNumber1 = 2 \* RandNumber3;  
 } else if (RandNumber3 == RandNumber2) {  
 RandNumber1 = RandNumber2;  
 RandNumber2 = RandNumber3;  
 } else {  
 continue;  
 }  
 System.*out*.println(RandNumber1);  
 }  
 //Task 3  
/\* System.out.println("Task 3");  
 RandNumber1 = RandNumber2 = 0;  
 for(;true; ){  
 int RandNumber3 = random.nextInt(100);  
 if(RandNumber3 > RandNumber2){  
 RandNumber1 = 2 \* RandNumber3;  
 }  
 else if(RandNumber3 == RandNumber2){  
 RandNumber1 = RandNumber2;  
 RandNumber2 = RandNumber3;  
 }  
 else {  
 continue;  
 }  
 System.out.println(RandNumber1);  
 }\*/  
  
 //Task 4  
 System.*out*.println("Task 4");  
 int numberOfPrimes = 25;  
 for (int primeNumbersCounter = 0, k = 1; primeNumbersCounter < (numberOfPrimes - 1); ) {  
 int numberOfDividers = 0;  
 for (int i = k / 2; i > 0; i--) {  
 if (k % i == 0) {  
 numberOfDividers++;  
 }  
 }  
 if (numberOfDividers == 2) {  
 int z = (int) Math.*sqrt*(k);  
 System.*out*.println(z);  
 primeNumbersCounter++;  
 }  
 k++;  
 }  
 //Task 5  
 System.*out*.println("Task 5");  
 int Val1 = 0xAA, Val2 = 0x55;  
  
 System.*out*.println(*toBinaryString*(Val1));  
 System.*out*.println(*toBinaryString*(Val2));  
 System.*out*.println("Val1 & Val2 = " + *toBinaryString*(Val1 & Val2));  
 System.*out*.println("Val1 | Val2 = " + *toBinaryString*(Val1 | Val2));  
 System.*out*.println("Val1 ^ Val2 = " + *toBinaryString*(Val1 ^ Val2));  
 System.*out*.println("~Val1 = " + *toBinaryString*(~Val1));  
 System.*out*.println("~Val2 = " + *toBinaryString*(~Val2));  
 //Task 6  
 System.*out*.println("Task 6");  
 System.*out*.println("test(10,1,6) = " + *test*(10, 1, 6));  
 System.*out*.println("test(5,1,6) = " + *test*(5, 1, 6));  
 System.*out*.println("test(6,1,6) = " + *test*(6, 1, 6));  
 //Task 7  
 System.*out*.println("Task 7");  
 for (int i = 1; i <= 100; i++) {  
 System.*out*.println(i);  
 if (i == 99) {  
 break;  
 }  
 }  
 /\* for (int i = 1; i <= 100; i++) {  
 System.out.println(i);  
 if(i ==99){  
 return;  
 }  
 }  
 \*/  
 //Task 8  
 System.*out*.println("Task 8");  
 System.*out*.println("1) With breaks");  
 for(int i =0; i< 3; i++){  
 switch (i){  
 case 0:  
 System.*out*.println(i);  
 break;  
 case 1:  
 System.*out*.println(i\*2);  
 break;  
 default:  
 System.*out*.println(i\*3);  
 }  
 }  
 System.*out*.println("2) Without breaks");  
 for(int i =0; i< 3; i++){  
 switch (i){  
 case 0:  
 System.*out*.println(i);  
 case 1:  
 System.*out*.println(i\*2);  
 default:  
 System.*out*.println(i\*3);  
 }  
 }  
 //Task 9  
 System.*out*.println("Tssk 9");  
 Fibonacci.*printFibonacci*(100);  
 }  
  
 private static String toBinaryString(int value) {  
 String result = "";  
 int copy = value;  
 do {  
 result = (copy % 2 != 0) ? 1 + result : 0 + result;  
 copy >>>= 1;  
 } while (copy != 0);  
 return result;  
 }  
  
 static int test(int testValue, int begin, int end) {  
 if (testValue > begin && testValue < end) {  
 return +1;  
 }  
 if (testValue == begin || testValue == end) {  
 return 0;  
 }  
 return -1;  
  
 }  
  
}  
  
class Fibonacci {  
 static void printFibonacci(int bound) {  
 int F1 =1, F2 = 1;  
 int currentValue = F1;  
 for(int i = 0; currentValue < bound; i++){  
 System.*out*.println(currentValue);  
 if(i<2){  
 continue;  
 }  
 currentValue = F1 + F2;  
 F1 = F2;  
 F2 = currentValue;  
 }  
 }  
}

**Реалізація програми:**

Connected to the target VM, address: '127.0.0.1:57066', transport: 'socket'

Task 1

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

Task 2

192

4

18

6

12

196

186

76

118

168

44

186

2

104

54

74

6

196

110

90

178

8

158

80

118

Task 4

2

3

5

7

11

13

17

19

23

29

31

37

41

43

47

53

59

61

67

71

73

79

83

89

Task 5

10101010

1010101

Val1 & Val2 = 0

Val1 | Val2 = 11111111

Val1 ^ Val2 = 11111111

~Val1 = 11111111111111111111111101010101

~Val2 = 11111111111111111111111110101010

Task 6

test(10,1,6) = -1

test(5,1,6) = 1

test(6,1,6) = 0

Task 7

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

Task 8

1) With breaks

0

2

6

2) Without breaks

0

0

0

2

3

6

Tssk 9

1

1

1

2

3

5

8

13

21

34

55

89

Disconnected from the target VM, address: '127.0.0.1:57066', transport: 'socket'

Process finished with exit code 0