

Laboratory work #2 | Control-flow basics

Lab objectives

As a result of this laboratory work you will extend your skills in Java programming:

- using of Eclipse integrated development environment (IDE);
- using of primitive data types, variables and constants.
- how to organize the control flow of your program: using conditional operators (if-else), loops (for, while).
- how to format the output of your program.

Overview

The goal of this task is calculating the values of a function on the range of arguments: $f(x)$, $x \in [a; b]$. You should calculate these values with step dx . For example, begin computation from the a , and change the argument by adding dx ($a + dx$), while it is less than b .

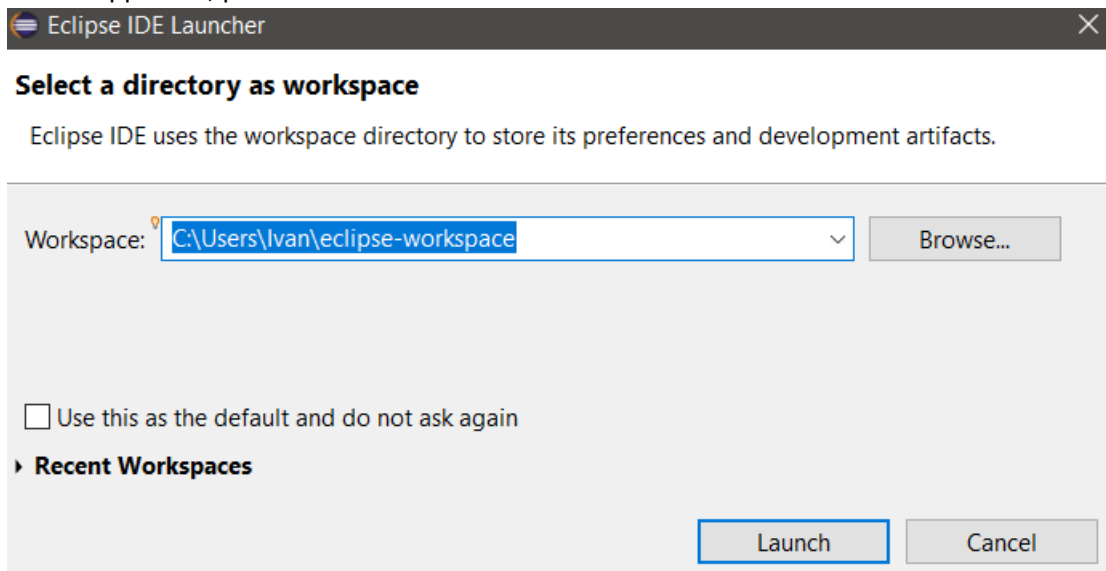
Task

1. Run the Eclipse integrated development environment (IDE).
2. Get acquainted with your variant. The variants you can find at the end of this document.
3. Implement the calculation of the function's value as the separate method which gets x as an argument.
4. Create a method to read double values from the keyboard.
5. Input the set of values.
6. Check the program correctness by calculating some $f(x)$ by hands (in piece of paper).

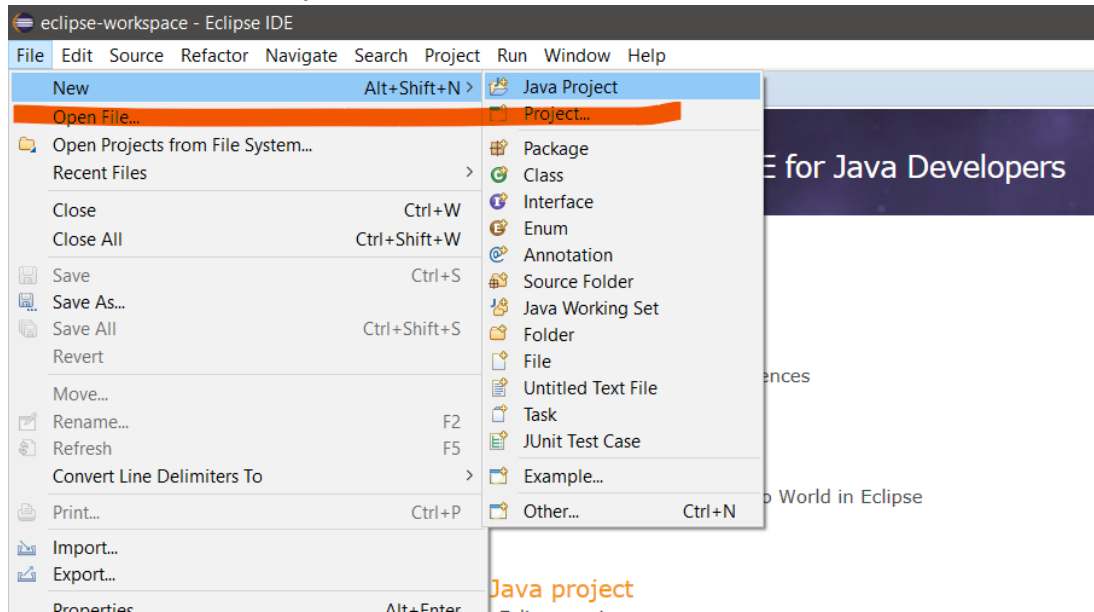
Instructions

1. Run the Eclipse integrated development environment (IDE). Create the new Java Project.

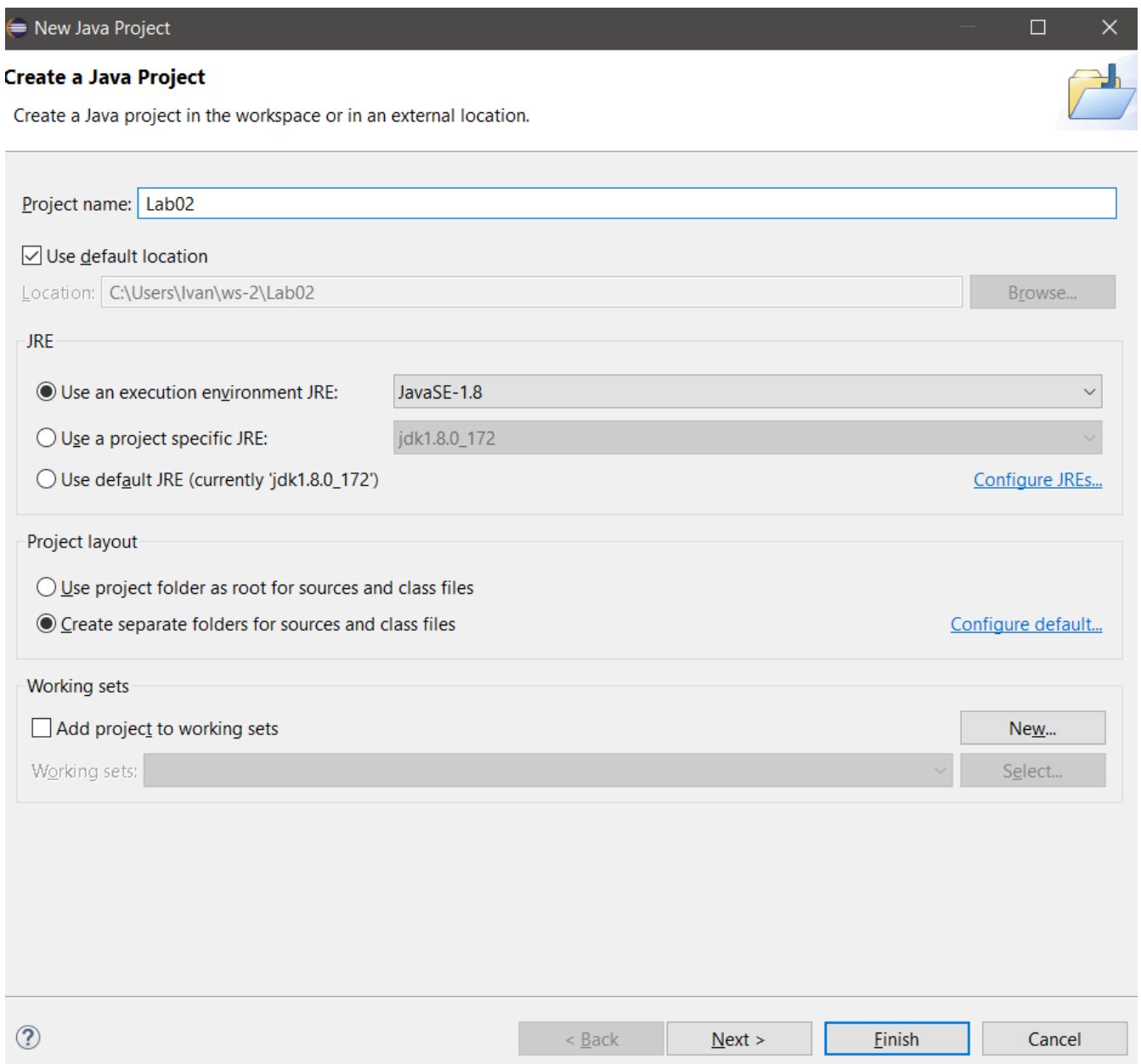
- 1.1. If this window appeared, press the "Launch" button.



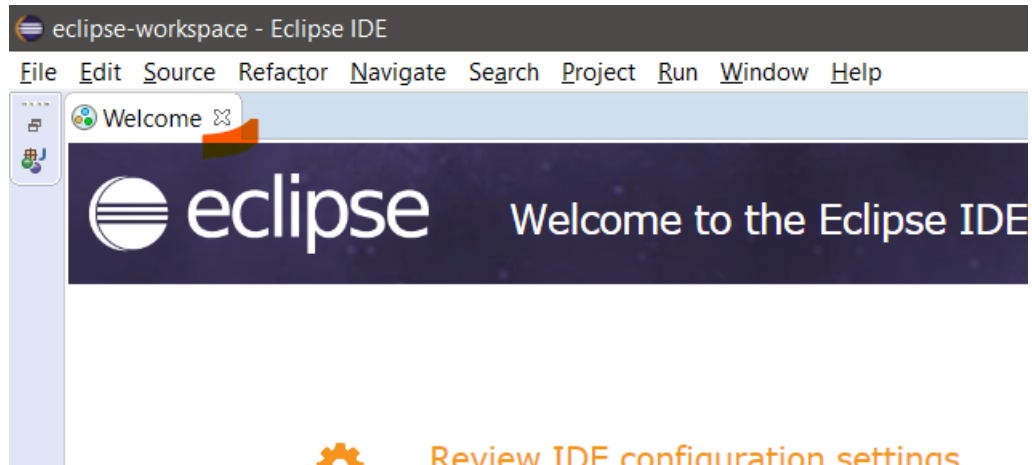
1.2. Choose the “File / New / Java Project” menu item.



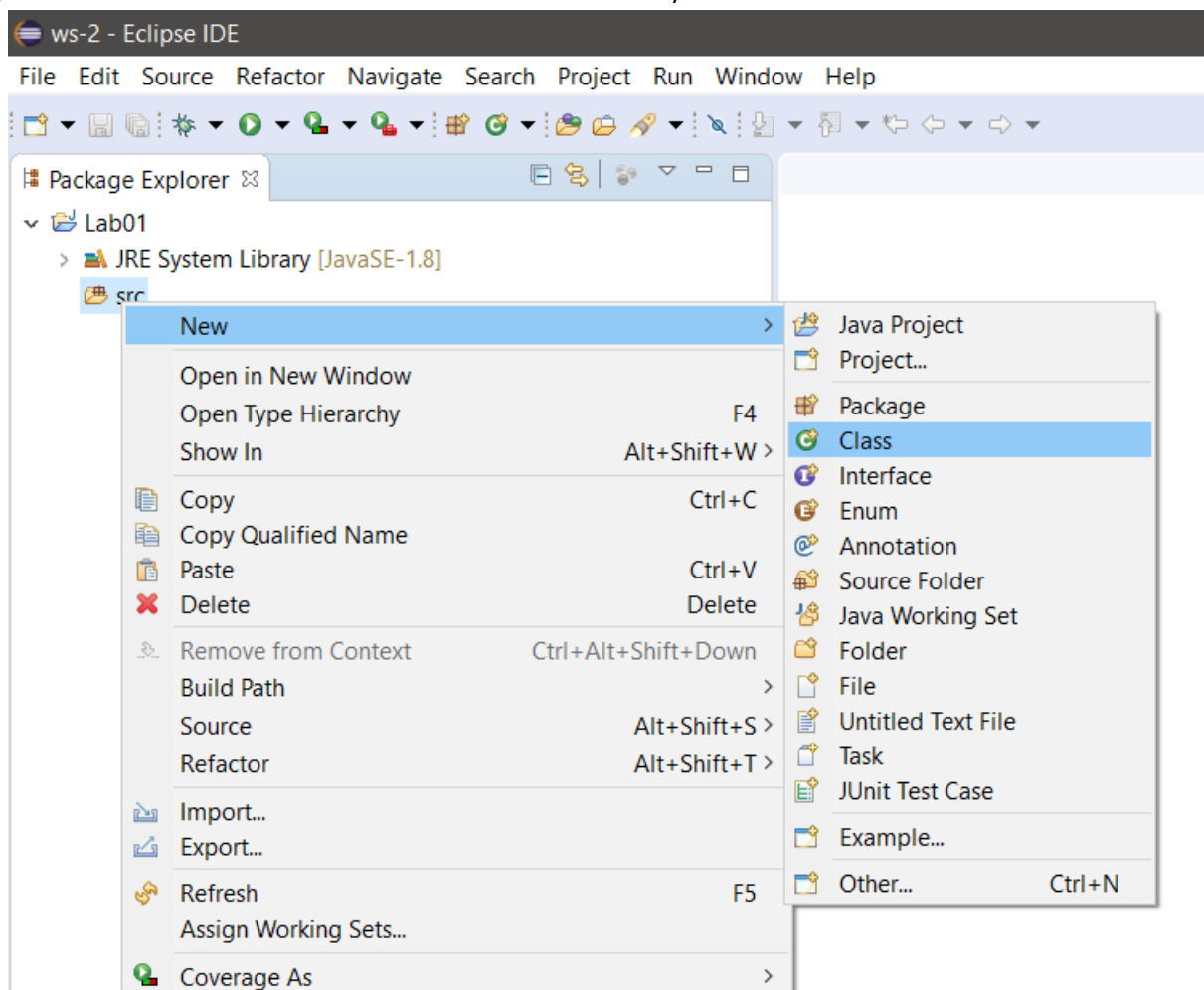
1.3. Write the name of a project – for example, Lab02, and press “Finish” button.



- 1.4. You will see your workspace (as shown in the screenshot below). If you can't see this, close the Welcome-page (press the close button near in tab, see the next screenshot).



- 1.5. Right-click on the folder "src" as shown below. Select "New / Class" menu item.



- 1.6. Enter the name of the class and check those checkboxes in state like at the screenshot below. Press the "Finish" button.

The screenshot shows the 'New Java Project' dialog box. The 'Project name' field is filled with 'Lab02'. The 'Use default location' checkbox is checked, and the 'Location' field shows 'C:\Users\Ivan\ws-2\Lab02'. Under the 'JRE' section, 'Use an execution environment JRE' is selected, with 'JavaSE-1.8' chosen from the dropdown. The 'Project layout' section has 'Create separate folders for sources and class files' selected. The 'Working sets' section has 'Add project to working sets' unchecked. At the bottom, the 'Finish' button is highlighted.

New Java Project

Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location
Location: [Browse...](#)

JRE

☒ Use an execution environment JRE: [Configure JREs...](#)

☐ Use a project specific JRE:

☐ Use default JRE (currently 'jdk1.8.0_172')

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

[? < Back](#) [Next >](#) **Finish** [Cancel](#)

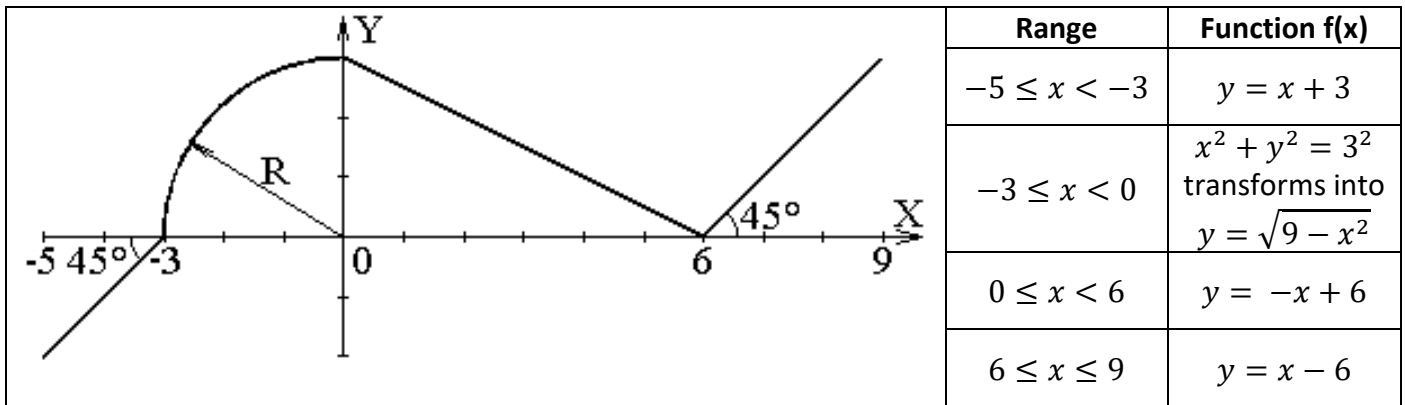
- 1.7. Now you can write code.

The screenshot shows the Eclipse IDE with the 'Lab02.java' file open. The Package Explorer on the left shows the project structure: 'Lab01' (containing 'JRE System Library [JavaSE-1.8]' and 'src'), and 'Lab02' (containing 'JRE System Library [JavaSE-1.8]', 'src', and '(default package)' which contains 'Lab02.java'). The main editor shows the code for 'Lab02.java'.

```
1  
2 public class Lab02 {  
3  
4     public static void main(String[] args) {  
5  
6  
7     }  
8  
9 }  
10
```

2. Get acquainted with your variant. The variants you can find at the end of this document.

2.1. For example, your variant defines this function:



It means that when a user entering value of X, program will use the function that corresponds for the range of X.

3. Implement the calculation of the function's value as the separate method which gets x as an argument.

- 3.1. Create the new method, `getValue` that accepts parameter `x` of double data type, and returns double value as a result.
- 3.2. This method implements checking of the range – where is X? Depends on this value, the program will calculate Y for the corresponding part of the function (look at the graphical representation of this function).

```
public static double getValue(double x) {
    //not a number
    //indicates that we have no value
    //(not calculated yet)
    double result = Double.NaN;

    if (x >= -5 && x < -3) {
        result = x + 3;
    }
    else if (x >= -3 && x < 0) {
        result = Math.sqrt(9 - x * x);
    }
    else if (x >= 0 && x < 6) {
        result = -x + 6;
    }
    else if (x >= 6 && x <= 9) {
        result = x - 6;
    }

    //in case if we have X out of range,
    //Not a Number value will be returned.
    //It means that we can't calculate value of the function for this X
    return result;
}
```

- 3.3. In the main method we call this `getValue` method twice to check it. Compile your program and look at the results.

4. Create a method to read double values from the keyboard.

4.1. If your program can't get data from the user, it could be useless. This is the reason why we will create the method that will read data from the console (terminal).

```
public static double readDouble(String message) {
    //print the greeting message
    System.out.print(message);

    //create scanner for standard input stream (keyboard)
    Scanner scanner = new Scanner(System.in);

    //read the value
    double val = scanner.nextDouble();

    //return result
    return val;
}
```

4.2. Use these methods together!

```
public static void main(String[] args) {
    double x = readDouble("Enter the x coordinate: ");
    double y = getValue(x);
    System.out.println("Y = " + y);
}
```

5. Input the set of values.

To make it possible, we need to organize repeating of users's input and output of the result. To do it, we will use loops.

```
public static void main(String[] args) {
    System.out.println("Values of the function will be calculated 5 times.");
    for (int i = 0; i < 5; i++) {
        System.out.print("#" + (i + 1) + ". ");
        double x = readDouble("Enter the x coordinate: ");
        double y = getValue(x);
        System.out.println("Y = " + y);
    }
}
```

But what if you want to get more detailed results? We will do this job:

- 5.1. Input the minimum and maximum values of the range. In this range our program will try to get value of the function $y = f(x)$. (Code on the next page).
- 5.2. We call the readDouble method again, to read the min and max values of the range from the keyboard.
- 5.3. Also, we have modified the "for" loop to iterate not N times from the zero to N, but from minimal value in our range to maximum. It will be done with the step with size 1 (changing loop's counter by one).

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

    double min = readDouble("Enter the minimum value of the range of X: ");
    double max = readDouble("Enter the maximum value of the range of X: ");

    for (double x = min; x < max; x++) {
        double y = getValue(x);
        System.out.println("X = " + x + "; Y = " + y);
    }
}
```

- 5.4. But the fixed step isn't interesting: what if user want to do more exact computations? Let's add dx variable (delta of x – or step – by this value will be changed current X to calculate our function's value $y = f(x)$).

```
public static void main(String[] args) {
    double min = readDouble("Enter the minimum value of the range of X: ");
    double max = readDouble("Enter the maximum value of the range of X: ");
    double dx = readDouble("Enter the step (dx): ");

    for (double x = min; x < max; x += dx) {
        double y = getValue(x);
        System.out.println("X = " + x + "; Y = " + y);
    }
}
```

Output of the program will be like this:

```
Enter the minimum value of the range of X: -7
Enter the maximum value of the range of X: 10
Enter the step (dx): 1,5
X = -7.0; Y = NaN
X = -5.5; Y = NaN
X = -4.0; Y = -1.0
X = -2.5; Y = 1.6583123951777
X = -1.0; Y = 2.8284271247461903
X = 0.5; Y = 5.5
X = 2.0; Y = 4.0
X = 3.5; Y = 2.5
X = 5.0; Y = 1.0
X = 6.5; Y = 0.5
X = 8.0; Y = 2.0
X = 9.5; Y = NaN= 8.0; Y = 2.0
```

It looks not well-structured, and the “NaN” strings isn't clear for the end-user. Let's think about formatting of the output. To fix the number of digits after the floating-point (dot) we can use the format string.

```
public static void main(String[] args) {
    double min = readDouble("Enter the minimum value of the range of X: ");
    double max = readDouble("Enter the maximum value of the range of X: ");
    double dx = readDouble("Enter the step (dx): ");

    for (double x = min; x < max; x += dx) {
        double y = getValue(x);
        System.out.println(String.format("Y = f(%.2f) = %.2f", x, y));
    }
}
```

And the output of this program:

```
Enter the minimum value of the range of X: -5
Enter the maximum value of the range of X: 5
Enter the step (dx): 1,5
Y = f(-5,00) = -2,00
Y = f(-3,50) = -0,50
Y = f(-2,00) = 2,24
Y = f(-0,50) = 2,96
Y = f(1,00) = 5,00
Y = f(2,50) = 3,50
Y = f(4,00) = 2,00
```

All code of this example:

```
import java.lang.Math;
import java.util.Scanner;

public class HelloWorld {

    public static double getValue(double x) {
        //not a number
        //indicates that we have no value
        //(not calculated yet)
        double result = Double.NaN;

        if (x >= -5 && x < -3) {
            result = x + 3;
        }
        else if (x >= -3 && x < 0) {
            result = Math.sqrt(9 - x * x);
        }
        else if (x >= 0 && x < 6) {
            result = -x + 6;
        }
        else if (x >= 6 && x <= 9) {
            result = x - 6;
        }

        //in case if we have X out of range,
        //Not a Number value will be returned.
        //It means that we can't calculate value of the function for this X
        return result;
    }

    public static double readDouble(String message) {
        //print the greeting message
        System.out.print(message);

        //create scanner for standard input stream (keyboard)
        Scanner scanner = new Scanner(System.in);
        //read the value
        double val = scanner.nextDouble();

        //return result
        return val;
    }

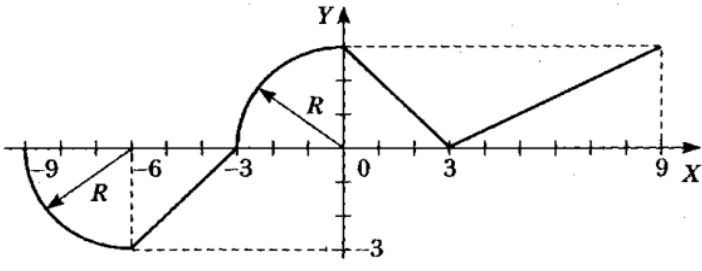
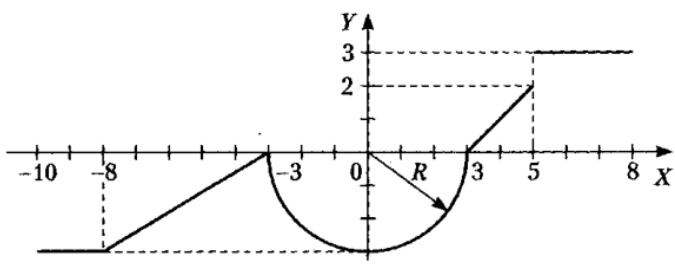
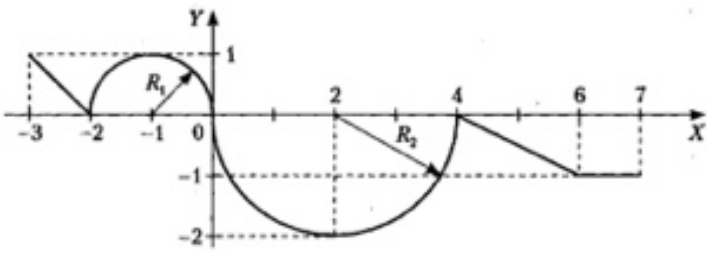
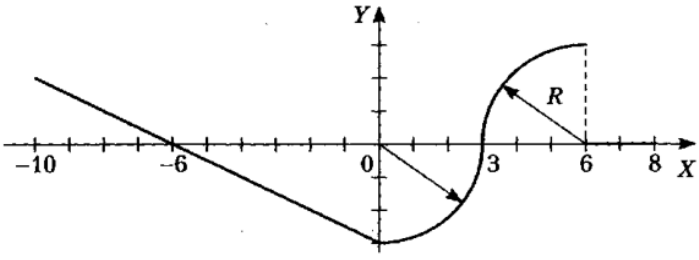
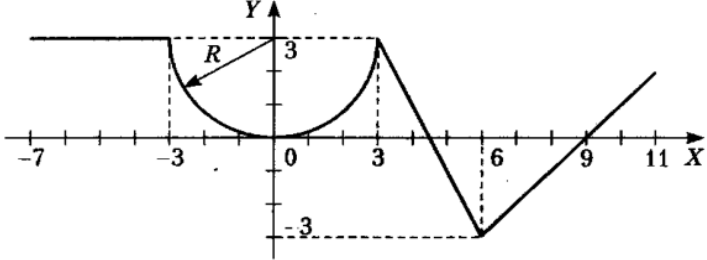
    public static void main(String[] args) {

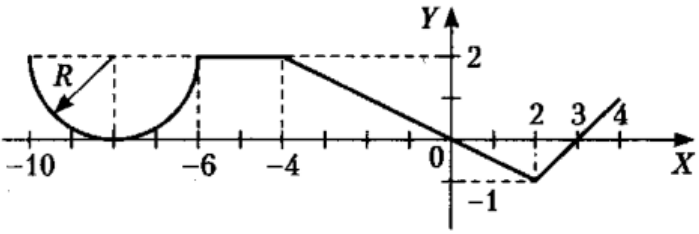
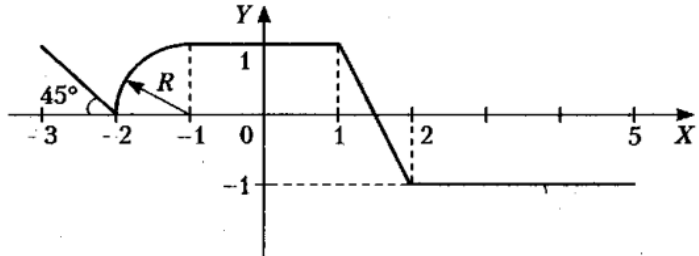
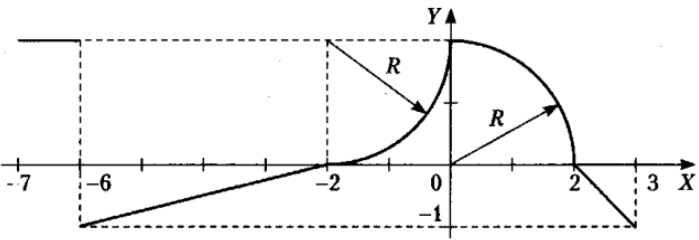
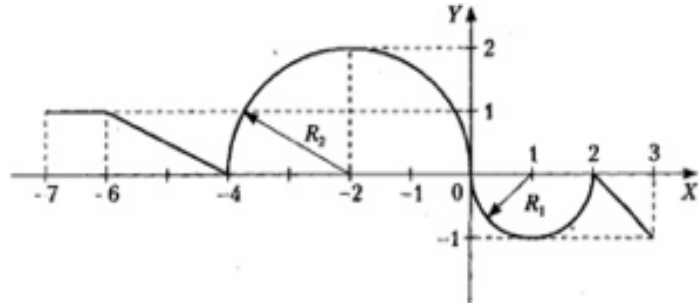
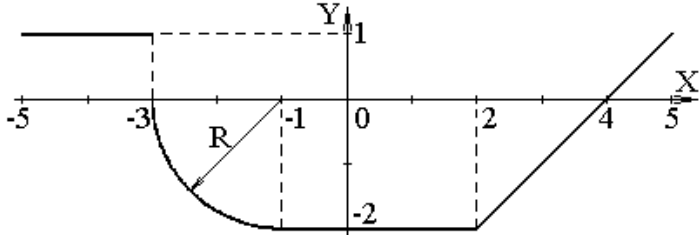
        double min = readDouble("Enter the minimum value of the range of X: ");
        double max = readDouble("Enter the maximum value of the range of X: ");
        double dx = readDouble("Enter the step (dx): ");

        for (double x = min; x < max; x += dx) {
            double y = getValue(x);
            System.out.println(String.format("Y = f(%.2f) = %.2f", x, y));
        }
    }
}
```

6. Check the program correctness by calculating some $f(x)$ by hands (in piece of paper).

Branching: Piecewise-defined function – Variants

1		Range	Function f(x)
		$-9 \leq x < -6$	$y = -\sqrt{9 - (x + 6)^2}$
		$-6 \leq x \leq -3$	$y = x + 3$
		$-3 < x < 0$	$y = \sqrt{9 - x^2}$
		$0 \leq x \leq 3$	$y = -x + 3$
2		Range	Function f(x)
		$-10 \leq x < -8$	$y = -3$
		$-8 \leq x \leq -3$	$y = 0.6 * x + 1.8$
		$-3 < x < 3$	$y = -\sqrt{9 - x^2}$
		$3 \leq x \leq 5$	$y = x - 3$
3		Range	Function f(x)
		$-3 \leq x \leq -2$	$y = -x - 2$
		$-2 < x < 0$	$y = \sqrt{1 - (x + 1)^2}$
		$0 \leq x \leq 4$	$y = -\sqrt{4 - (x - 2)^2}$
		$4 < x \leq 6$	$y = -0.5 * x + 2$
4		Range	Function f(x)
		$-10 \leq x \leq 0$	$y = -0.3 * x - 3$
		$0 < x < 3$	$y = -\sqrt{9 - x^2}$
		$3 \leq x \leq 6$	$y = \sqrt{9 - (x - 6)^2}$
5		Range	Function f(x)
		$-7 < x \leq -3$	$y = 3$
		$-3 < x \leq 3$	$y = \sqrt{9 - x^2} + 3$
		$3 < x \leq 6$	$y = -1.5 * x + 7.5$
		$6 < x \leq 9$	$y = x - 3$

6		Range	Function f(x)
		$-10 < x \leq -6$	$y = \sqrt{(x+8)^2} + 2$
		$-6 < x \leq -4$	$y = 2$
		$-4 < x \leq 2$	$y = -0.5 * x$
		$2 < x \leq 4$	$y = x - 3$
7		Range	Function f(x)
		$-3 < x \leq -2$	$y = -x - 2$
		$-2 < x \leq -1$	$y = \sqrt{1 - (x+1)^2}$
		$-1 < x \leq 1$	$y = 1$
		$1 < x \leq 2$	$y = -2 * x + 3$
8		Range	Function f(x)
		$-7 \leq x \leq -6$	$y = 2$
		$-6 < x \leq -2$	$y = 0.25 * x + 0.5$
		$-2 < x \leq 0$	$y = -\sqrt{4 - (x+2)^2} + 2$
		$0 < x \leq 2$	$y = \sqrt{4 - x^2}$
9		Range	Function f(x)
		$-7 \leq x \leq -6$	$y = 1$
		$-6 < x \leq -4$	$y = -0.5 * x - 2$
		$-4 < x < 0$	$y = \sqrt{4 - (x+4)^2}$
		$0 \leq x \leq 2$	$y = -\sqrt{1 - (x-1)^2}$
10		Range	Function f(x)
		$-5 \leq x \leq -3$	$y = 1$
		$-3 < x \leq -1$	$y = -\sqrt{4 - (x+1)^2}$
		$-1 < x \leq 2$	$y = -2$
		$2 \leq x \leq 5$	$y = x - 4$

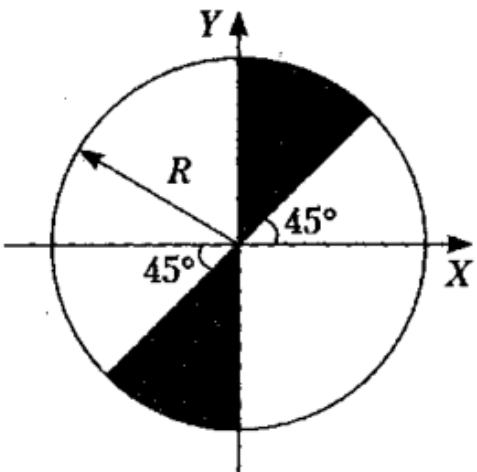
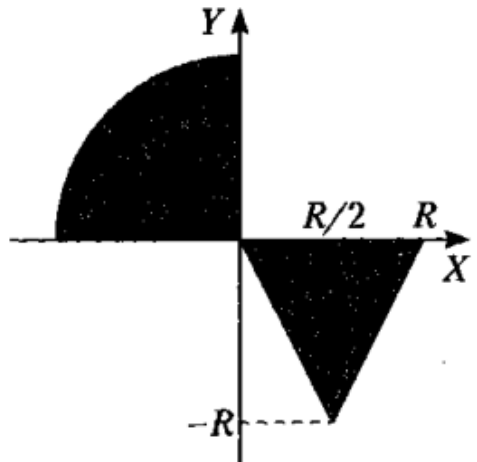
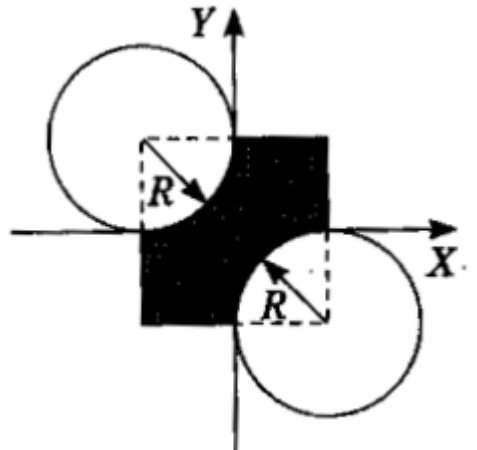
Extra tasks

The following tasks intended to provide more experience specifically on the topics about control-flow statements. They are not under assessment on the course.

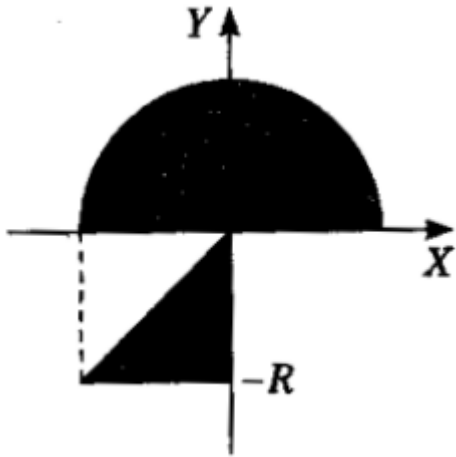
If you have any problems while being interested in accomplishing these tasks, some reasonable guidance still can be provided.

2.1 Branching: Hit point

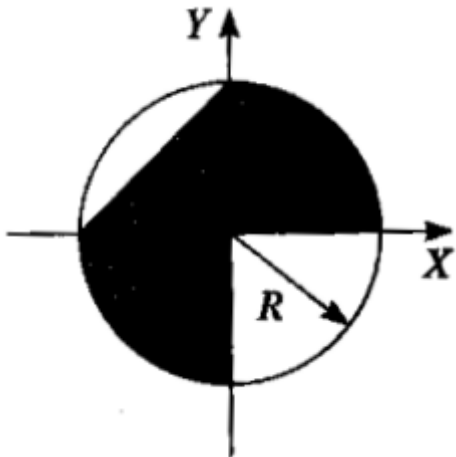
Write a program for hit point checking. The checking here means, wither given point is located inside of the area, represented with grayish part of a chart. All parameters, including point coordinates, should be read from standard input, result should be printed to standard output.

#	Description
1	
2	
3	

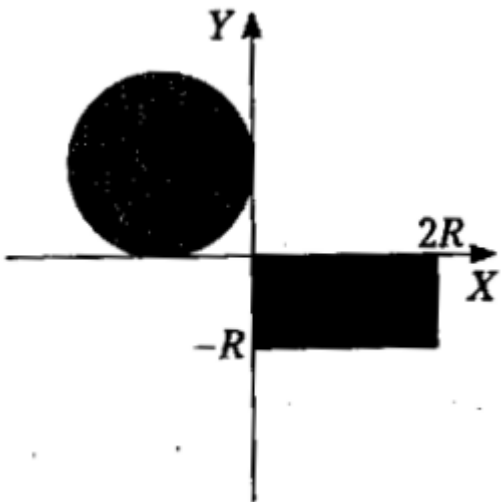
4



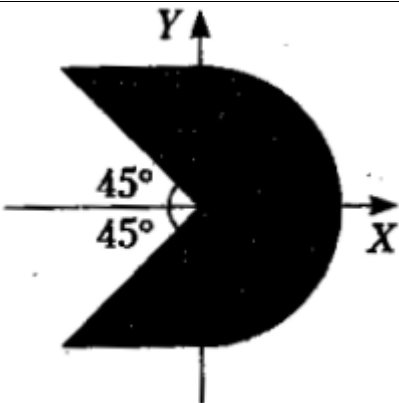
5

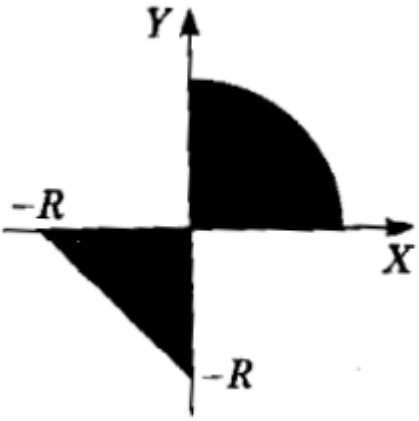
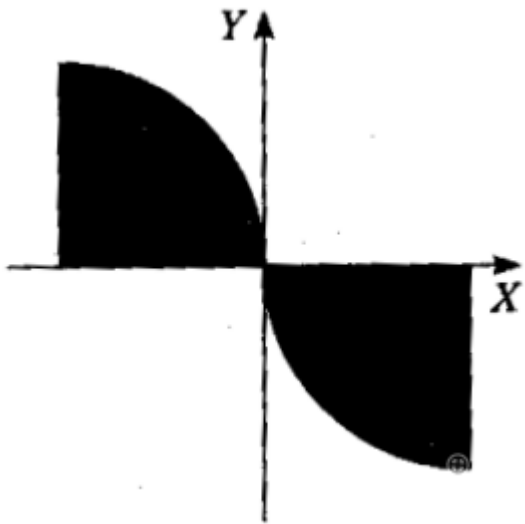
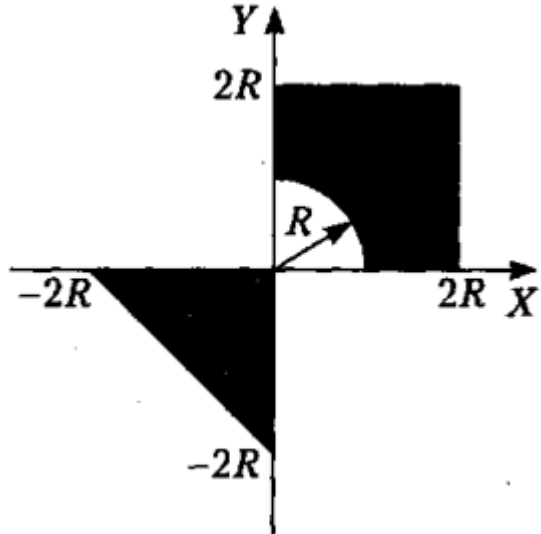


6



7



8	
9	
10	

2.2 Loops: Table of function values

Print to the standard output a series of values of the function from lab 2.1. Output should be presented as table with columns X and Y, where X is in range $[X1, X2]$ taken with step dX . Select $X1$, $X2$ and dX such that all branches pieces of function would be used during computation.

2.3 Loops: Taylor series

Compute and print to standard output values of the function using given Taylor series form. Output should be presented as table with columns X and Y, where X is in range [X1, X2] taken with step dX. Parameters to read from standard input are: X1, X2, dX, precision E. Here, precision means lower bound of Taylor series step size.

#	Description
1	$\ln(x+1) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{n+1}}{n+1} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} - \dots, \quad -1 < x \leq 1.$
2	$e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \dots, \quad x < \infty.$
3	$\ln \frac{1+x}{1-x} = 2 \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = 2 \left(x + \frac{x^3}{3} + \frac{x^5}{5} + \dots \right), \quad x < 1.$
4	$\operatorname{arctg} x = -\frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \dots, \quad x < -1.$
5	$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots, \quad x < \infty.$
6	$\frac{\sin x}{x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n+1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} - \dots, \quad x < \infty.$
7	$\ln x = \sum_{n=0}^{\infty} \frac{(-1)^n (x-1)^{n+1}}{(n+1)} = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} + \dots, \quad 0 < x \leq 2.$
8	$e^{-x^2} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots, \quad x < \infty.$
9	$\ln(x+1) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{n+1}}{n+1} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} - \dots, \quad -1 < x \leq 1.$
10	$\operatorname{arctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} x^{2n+1}}{2n+1} = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} - \dots, \quad x \leq 1.$