

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРА-
ЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра вычислительных технологий

ЛАБОРАТОРНАЯ РАБОТА №1
Дисциплина: Платформо-независимое программирование

Работу выполнил: _____ А. А. Костров

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

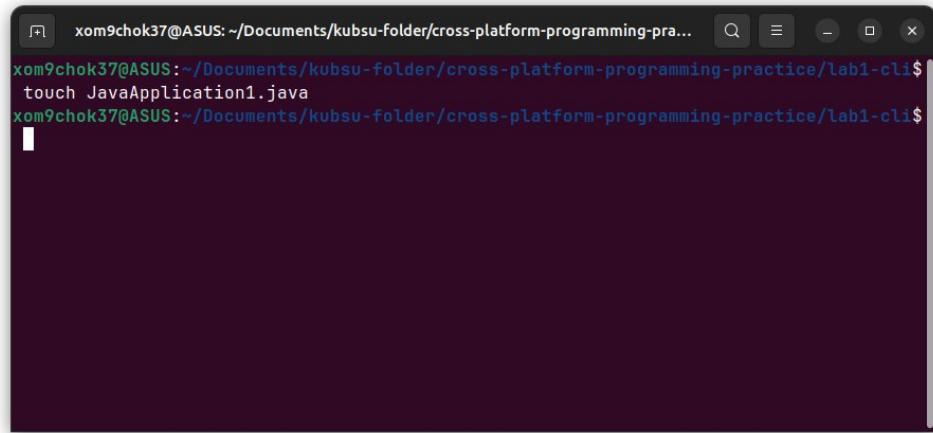
Преподаватель: _____ В. И. Шиян

Краснодар
2026

Тема. Работа в командной строке, компиляция и запуск на выполнение.

Цель. Освоить основы работы в командной строке Java. Изучить структуру и синтаксис простой программы. Посмотреть возможности языка.

1. Напишем, скомпилируем и выполним программу *Калькулятор* и примеры из командной строки (рисунки 1-14).



```
xom9chok37@ASUS: ~/Documents/kubsu-folder/cross-platform-programming-pra...
xom9chok37@ASUS:~/Documents/kubsu-folder/cross-platform-programming-practice/lab1-cli$ touch JavaApplication1.java
xom9chok37@ASUS:~/Documents/kubsu-folder/cross-platform-programming-practice/lab1-cli$
```

Рисунок 1 — Создадим файл *JavaApplication1.java*



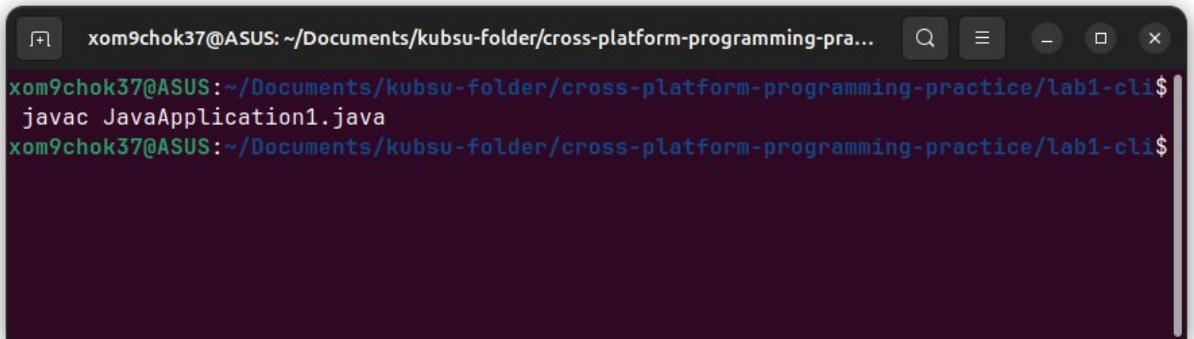
```
Open ▾ JavaApplication1.java ~/Documents/kubsu-folder/cross-platform-programming-practice/lab1-cli

public class JavaApplication1 {
    public static void main(String[] args) {
        System.out.println("Hello World!");
        Calculator calc=new Calculator();
        System.out.println("2+2="+calc.sum(2,2));
    }

    public static class Adder {
        private int sum;
        public Adder() { sum=0; }
        public Adder(int a) { this.sum=a; }
        public void add(int b) { sum+=b; }
        public int getSum() { return sum; }
    }

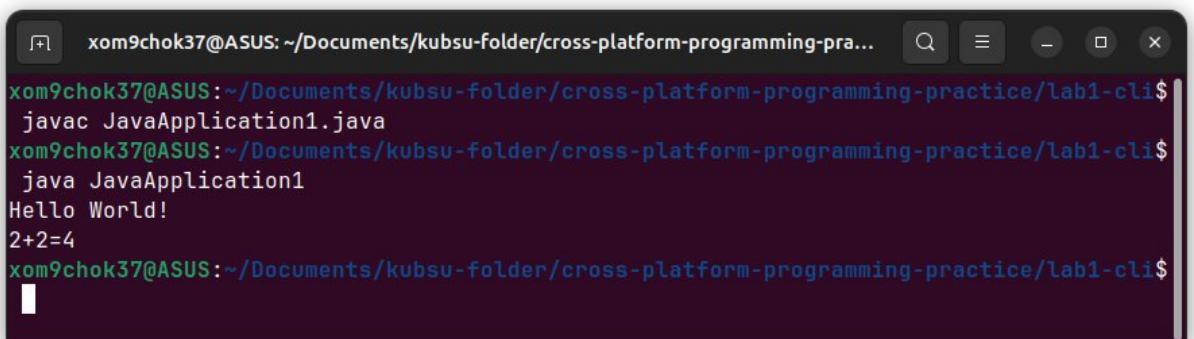
    public static class Calculator {
        public int sum(int... a) {
            Adder adder=new Adder();
            for (int i:a) {
                adder.add(i);
            }
            return adder.getSum();
        }
    }
}
```

Рисунок 2 — Напишем код программы *Калькулятор*



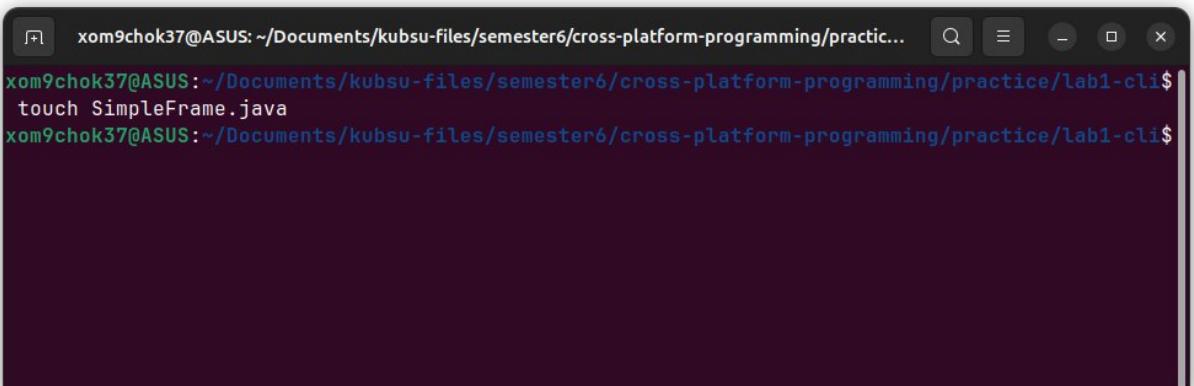
```
xom9chok37@ASUS: ~/Documents/kubsu-folder/cross-platform-programming-pra...$ javac JavaApplication1.java
xom9chok37@ASUS: ~/Documents/kubsu-folder/cross-platform-programming-practice/lab1-cli$
```

Рисунок 3 — Скомпилируем программу



```
xom9chok37@ASUS: ~/Documents/kubsu-folder/cross-platform-programming-pra...$ javac JavaApplication1.java
xom9chok37@ASUS: ~/Documents/kubsu-folder/cross-platform-programming-practice/lab1-cli$ java JavaApplication1
Hello World!
2+2=4
xom9chok37@ASUS: ~/Documents/kubsu-folder/cross-platform-programming-practice/lab1-cli$
```

Рисунок 4 — Выполним программу *Калькулятор*



```
xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practic...$ touch SimpleFrame.java
xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1-cli$
```

Рисунок 5 — Создадим файл *SimpleFrame.java*

The screenshot shows a Java code editor window titled "SimpleFrame.java". The code is written in Java and defines a class named "SimpleFrame" that extends the "Frame" class from the "java.awt" package. The code includes comments in Russian explaining the purpose of various methods and variables. It sets the window size to 400x150 pixels, makes it visible, and adds two window listeners. One listener handles the closing of the window by disposing of it and exiting the program. The other listener handles the closing of the window by disposing of it and exiting the program. The main method creates an instance of the SimpleFrame class with the title "Моя программа".

```
import java.awt.*;
import java.awt.event.*;

// Создаем наш класс, наследуясь от стандартного класса Frame
class SimpleFrame extends Frame {
    SimpleFrame(String s) {
        super(s);
        // Передаем название окна в родительский класс
        setSize(400, 150); // Устанавливаем размер окна
        setVisible(true); // Окно становится видимым
        // Добавляем слушателя, который будет реагировать на кнопку
        // закрытия окна
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent ev) {
                dispose(); // Закрываем окно
                System.exit(0); // Выходим из программы
            }
        });
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent ev) {
                dispose();
                System.exit(0);
            }
        });
    }
    // Создаем экземпляр нашего класса в главном классе
    public static void main(String[] args) {
        new SimpleFrame(" Моя программа");
    }
}
```

Рисунок 6 — Напишем код программы

The screenshot shows a terminal window with a dark background and light-colored text. The user is at the command prompt "xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1-clis\$". They type "javac SimpleFrame.java" and press Enter. The terminal then displays the command again followed by a blank line, indicating the compilation process.

```
xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1-clis$ javac SimpleFrame.java
xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1-clis$
```

Рисунок 7 — Скомпилируем пример 1.1

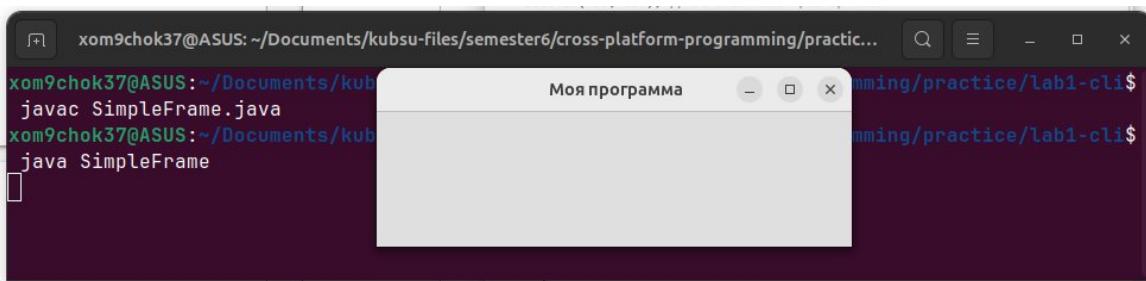


Рисунок 8 — Запустим пример 1.1

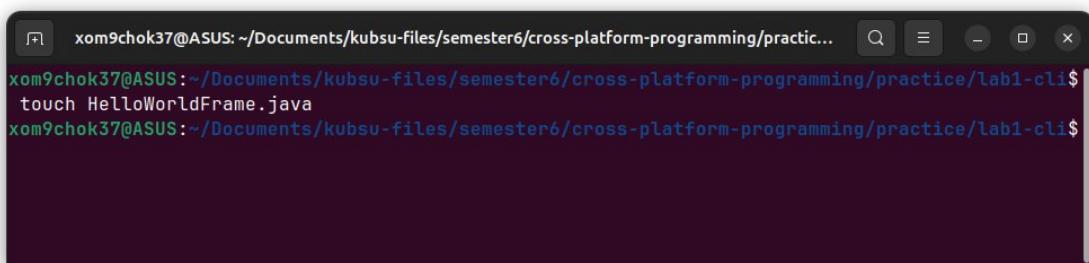


Рисунок 9 — Создадим файл *HelloWorldFrame.java*

```
Open ▾  ⌂ ~Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1-clli

// Подключаем библиотеку для работы с графикой
import java.awt.*;
import java.awt.event.*;

// Класс с полем для рисования
class HelloWorldFrame extends Frame {
    HelloWorldFrame(String s) {
        super(s);
    }
    public void paint(Graphics g) {
        g.setFont(new Font("Serif", Font.ITALIC | Font.BOLD, 30));
        g.drawString("Hello, XXI century World!", 20, 100);
    }
    public static void main(String[] args){
        Frame f = new HelloWorldFrame("Здравствуй, мир XXI века!");
        f.setSize(400, 150); //Задаем размер окна
        f.setVisible(true); //Делаем его видимым
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent ev) {
                System.exit(0);
            }
        });
    }
}
```

Рисунок 10 — Напишем код программы из примера 1.2

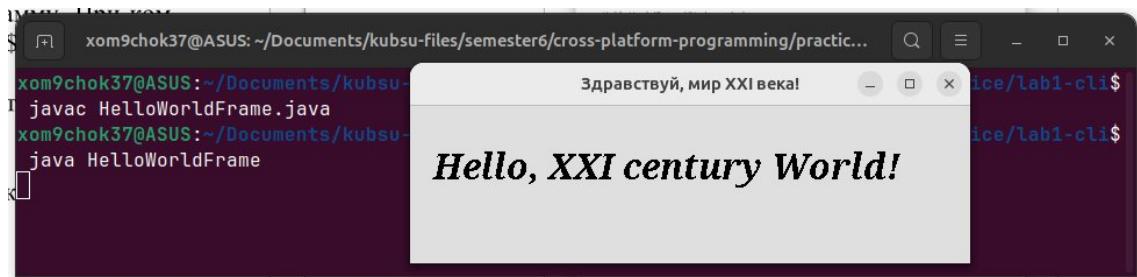


Рисунок 11 — Скомпилируем и запустим программу

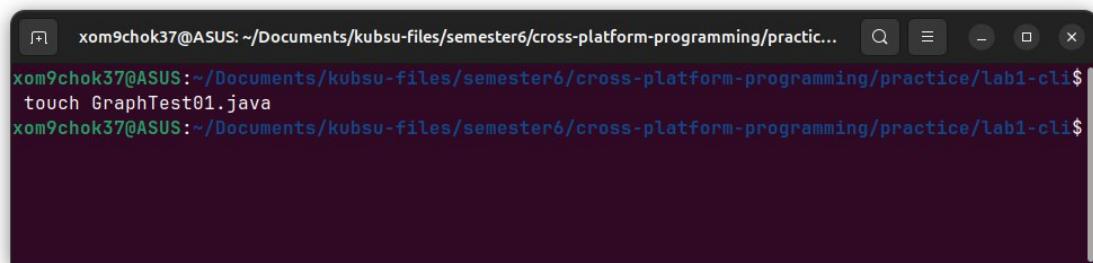


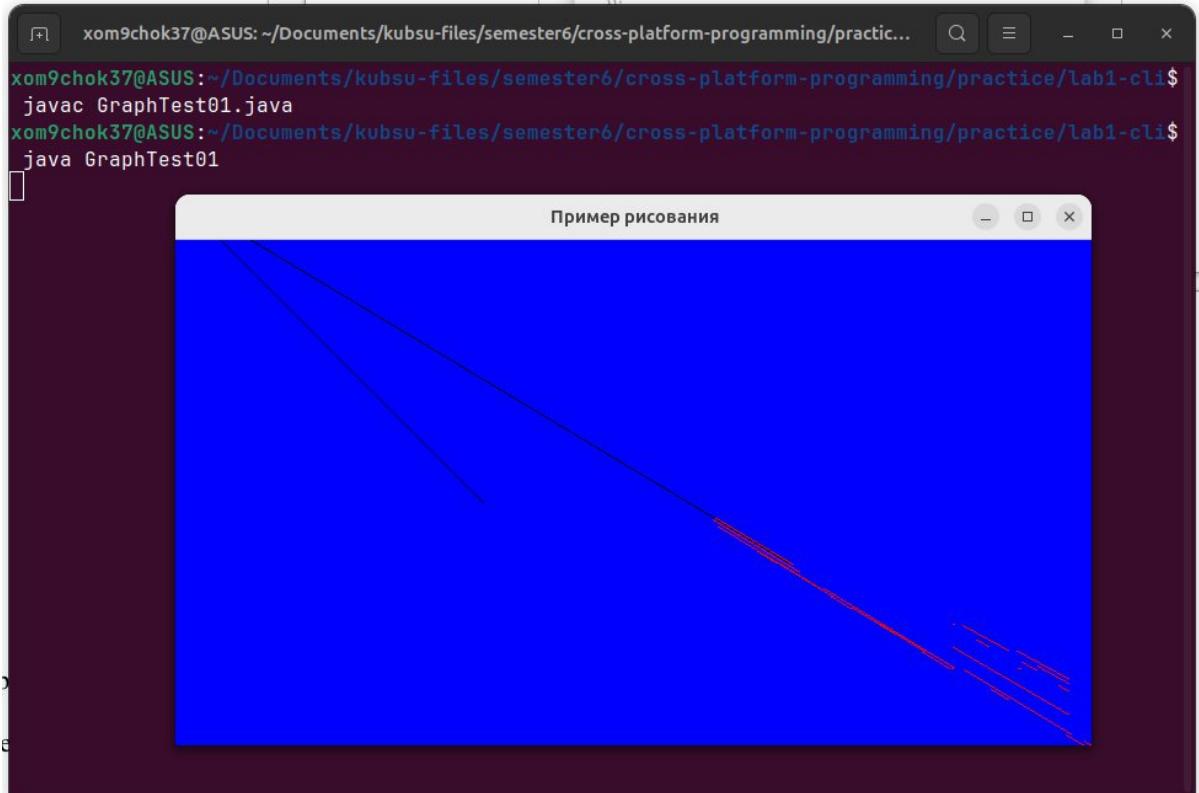
Рисунок 12 — Создадим файл *GraphTest01.java*

A screenshot of a code editor window titled 'GraphTest01.java'. The code is as follows:

```
import java.awt.*;
import java.awt.event.*;

class GraphTest01 extends Frame {
    GraphTest01(String s) {
        super(s);
        setBounds(0, 0, 500, 300);
        setVisible(true);
    }
    public void paint(Graphics g){
        Dimension d = getSize();
        int dx = d.width / 20, dy = d.height / 20;
        int myWidth = 250, myHeight = 250;
        g.drawLine(0, 0, myWidth, myHeight);
        g.drawLine(0, 0, d.width, d.height);
        setBackground(Color.blue);
        setForeground(Color.red);
    }
    public static void main(String[] args) {
        GraphTest01 f = new GraphTest01(" Пример рисования");
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent ev) {
                System.exit(0);
            }
        });
    }
}
```

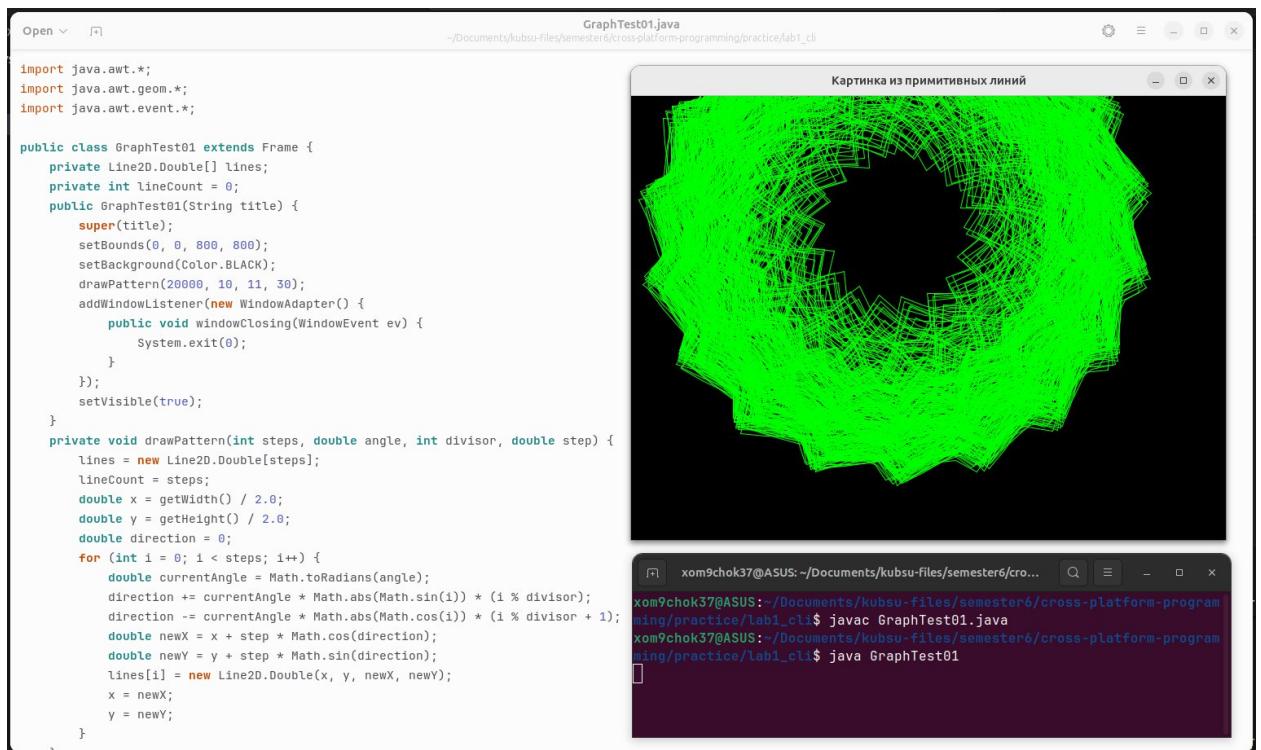
Рисунок 13 — Напишем код программы из примера 1.3



```
xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practic...$ javac GraphTest01.java
xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practic...$ java GraphTest01
[...]
Пример рисования
```

Рисунок 14 — Скомпилируем и запустим программу

2. Напишем программу, рисующую картинку из примитивных фигур (рисунок 15).



```
GraphTest01.java
```

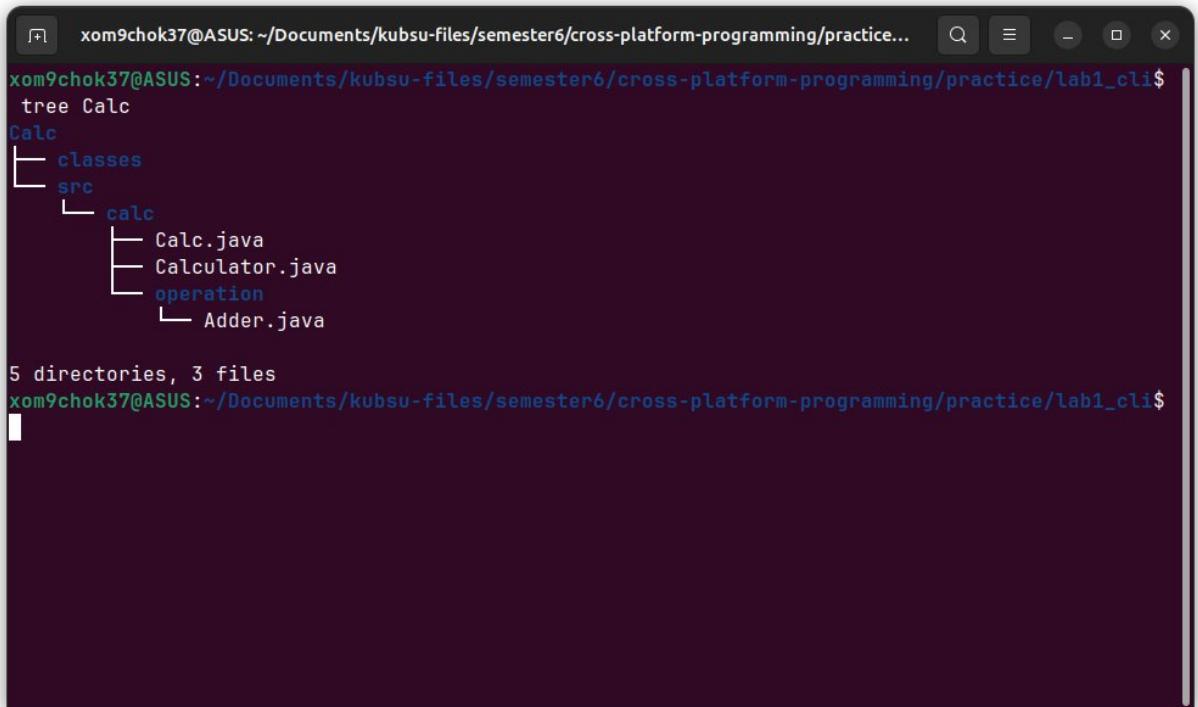
```
import java.awt.*;
import java.awt.geom.*;
import java.awt.event.*;

public class GraphTest01 extends Frame {
    private Line2D.Double[] lines;
    private int lineCount = 0;
    public GraphTest01(String title) {
        super(title);
        setBounds(0, 0, 800, 800);
        setBackground(Color.BLACK);
        drawPattern(20000, 10, 11, 30);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent ev) {
                System.exit(0);
            }
        });
        setVisible(true);
    }
    private void drawPattern(int steps, double angle, int divisor, double step) {
        lines = new Line2D.Double[steps];
        lineCount = steps;
        double x = getWidth() / 2.0;
        double y = getHeight() / 2.0;
        double direction = 0;
        for (int i = 0; i < steps; i++) {
            double currentAngle = Math.toRadians(angle);
            direction += currentAngle * Math.abs(Math.sin(i)) * (i % divisor);
            direction -= currentAngle * Math.abs(Math.cos(i)) * (i % divisor + 1);
            double newX = x + step * Math.cos(direction);
            double newY = y + step * Math.sin(direction);
            lines[i] = new Line2D.Double(x, y, newX, newY);
            x = newX;
            y = newY;
        }
    }
}
```

```
xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practic...$ javac GraphTest01.java
xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practic...$ java GraphTest01
```

Рисунок 15 — Скомпилируем и запустим программу

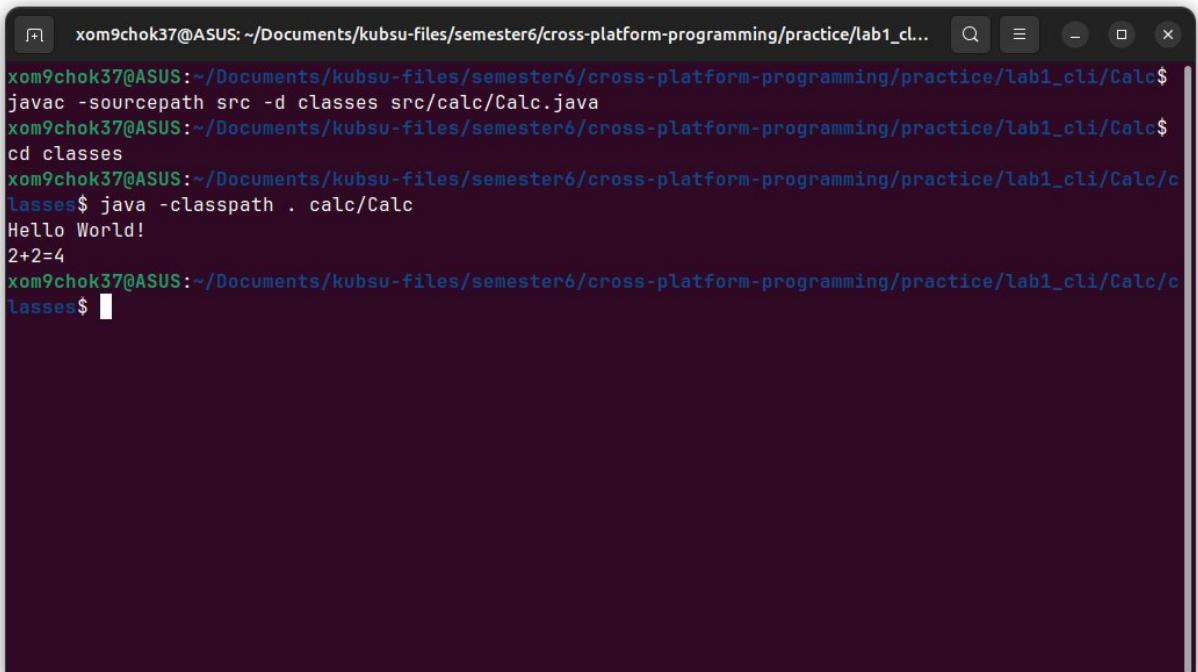
3. Создаём структуру приложения и компилируем (рисунки 16-17).



```
xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practice...$ tree Calc
Calc
└── classes
└── src
    └── calc
        ├── Calc.java
        ├── Calculator.java
        └── operation
            └── Adder.java

5 directories, 3 files
```

Рисунок 16 — Строим структуру



```
xom9chok37@ASUS: ~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc$ javac -sourcepath src -d classes src/calc/Calc.java
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc$ cd classes
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc/classes$ java -classpath . calc/Calc
Hello World!
2+2=4
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc/classes$
```

Рисунок 17 — Скомпилируем и запустим программу

4. Добавляем дополнительные классы, скомпилируем и запустим программу (рисунки 18-20).

```
Calc.java
package calc;

public class Calc {
    public static void main(String[] args) {
        System.out.println("Hello World!");
        Calculator calc=new Calculator();
        System.out.println("2+2="+calc.sum(2,2));
    }
}

Adder.java
package calc.operation;

public class Adder {
    private int sum;
    public Adder() { sum=0; }
    public Adder(int a) { this.sum=a; }
    public void add(int b) { sum+=b; }
    public int getSum() { return sum; }
}

Multer.java
public void multiply(int b) { mul*=b; }
public int getMul() { return mul; }

Subber.java
package calc.operation;

public class Subber {
    private int sub;
    public Subber() { sub=0; }
    public Subber(int a) { this.sub=a; }
    public void subtract(int b) { sub-=b; }
    public int getSub() { return sub; }
}

Diver.java
package calc.operation;

public class Diver {
    private int div;
    public Diver() { div=0; }
    public Diver(int a) { this.div=a; }
    public void divide(int b) { div/=b; }
    public int getDiv() { return div; }
}

Mover.java
package calc.operation;

public class Mover {
    private short mov;
    public Mover() { mov=0; }
    public Mover(short a) { this.mov=a; }
    public void move() { long tmp = mov; tmp <= 1; mov = (short)tmp; }
    public int getMov() { return mov; }
}
```

Рисунок 18 — Создаём новые классы

```
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc$ javac -sourcepath src -d classes src/calc/Calc.java
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc$ ls classes/calc/operation
Adder.class Diver.class Mover.class Multer.class Subber.class
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc$
```

Рисунок 19 — Скомпилируем программу

```
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc$ ls classes/calc/operation
Adder.class Diver.class Mover.class Multer.class Subber.class
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc$ cd classes
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc$ java -classpath . calc/Calc
Hello World!
2+2=4
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli/Calc$
```

Рисунок 20 — Запустим программу

5. Добавим манифест в корневой каталог нашей программы, затем соберём программу в *JAR* и запустим её (рисунок 21).

xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli\$ echo -e "Main-Class: calc.Calc\n" > Calc/MANIFEST.MF
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli\$ jar cvfm Calc.jar Calc/MANIFEST.MF -C Calc/classes/ .
added manifest
adding: calc/(in = 0) (out= 0)(stored 0%)
adding: calc/Calc.class(in = 945) (out= 548)(deflated 42%)
adding: calc/Calculator.class(in = 449) (out= 353)(deflated 21%)
adding: calc/operation/(in = 0) (out= 0)(stored 0%)
adding: calc/operation/Adder.class(in = 400) (out= 265)(deflated 33%)
adding: calc/operation/Diver.class(in = 403) (out= 265)(deflated 34%)
adding: calc/operation/Mover.class(in = 408) (out= 274)(deflated 32%)
adding: calc/operation/Multer.class(in = 407) (out= 271)(deflated 33%)
adding: calc/operation/Subber.class(in = 407) (out= 266)(deflated 34%)
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli\$ java -jar Calc.jar
Hello World!
2+2=4
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli\$ stat Calc.jar
 File: Calc.jar
 Size: 3747 Blocks: 8 IO Block: 4096 regular file
Device: 259,2 Inode: 25690120 Links: 1
Access: (0664/-rw-rw-r--) Uid: (1000/xom9chok37) Gid: (1000/xom9chok37)
Access: 2026-02-10 21:43:08.588585662 +0300
Modify: 2026-02-10 21:42:55.629320599 +0300
Change: 2026-02-10 21:42:55.630320620 +0300
 Birth: 2026-02-10 21:42:55.613320272 +0300
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli\$ tree Calc
Calc
├── classes
│ └── calc
│ ├── Calc.class
│ ├── Calculator.class
│ └── operation
│ ├── Adder.class
│ ├── Diver.class
│ ├── Mover.class
│ ├── Multer.class
│ └── Subber.class
└── MANIFEST.MF
src
└── calc
 ├── Calc.java
 ├── Calculator.java
 └── operation
 ├── Adder.java
 ├── Diver.java
 ├── Mover.java
 ├── Multer.java
 └── Subber.java

7 directories, 15 files
xom9chok37@ASUS:~/Documents/kubsu-files/semester6/cross-platform-programming/practice/lab1_cli

Рисунок 21 — Создаём манифест и собираем программу