DE Computational Practicum
Report

Student's name: Salikhova Elvira
Group: BS17-08
Variant: 19
Innopolis University, 2018

[Click here to see code on GitHub](https://github.com/elvirkavena/DE-Assignment.git)
https://github.com/elvirkavena/DE-Assignment.git

Solution:

1. Solving equation:

$$y' = \sqrt{y} + 2y \qquad\qquad y(0) = 1;$$

Bernoulli's equation

ODE classification: first - order non-linear ordinary differential equation

$$\frac{y'}{\sqrt{y}} - 2\sqrt{y} = 2$$

Substitution:

$$z = \sqrt{y}; 2z' = \frac{y'}{\sqrt{y}}$$

$$2z' - 2z = 2;$$

Substitution:

$$z = uv; z' = u'v + uv';$$

$$u'v + uv' - uv = 1;$$

$$\begin{cases} v' = v \\ u'v = 1 \end{cases}$$

$$v' = v;$$

$$u'e^x = 1;$$

$$\frac{du}{dx} = e^{-x}; \frac{dv}{v} = dx;$$

Integrating:

$$\int \frac{dv}{v} = \int dx;$$

$$ln\,|v| = x; v = e^x;$$

Integrating:

$$\int \frac{du}{dx} = \int e^{-x};$$

$$u = -e^{-x} + C;$$

$z = (-e^{-x} + C)e^{x} = Ce^{x} - 1;$

$y = (Ce^{x} - 1)^2;$ - general solution.

$y = 0$ - also solution.

2. IVP:

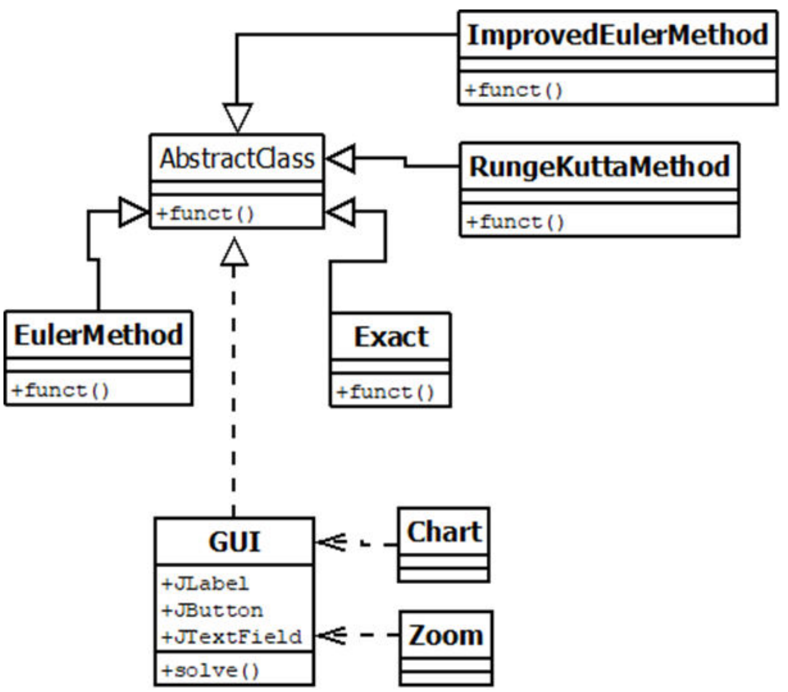
$y(0) = 1;$

$1 = (C - 1); C = 2;$

$y = (2e^{x} - 1)^2$ - particular solution

Code:

In main class, starting GUI class you can change values of step,x,y,and Xfinal.
There are 4 classes for each method :
Euler, Improved Euler Runge-Kutta and exact solution which are inherit abstract class and implements funct. In GUI class we solve numerical methods with given values, adding to chart. When program will calculate all values it shoes window with two chart - methods and errors of methods.



The UML diagram

Methods implementation:

1. Exact method

```
public Exact(){}

//Series of exact solution
XYSeries series1;

public void funct(Float step, Float x, Float y, Float xf){
    //Calculating points that satisfy to Exact solution and adding them to the series:
    series1 = new XYSeries( key: "Exact");
    float e = 2.71828f;
    for(float i = x; i <= xf; i += step) {
        double f = Math.pow((2*Math.pow(e,x)-1),2);
        series1.add(i, f);
    }
}
```

2. Euler method

```
public class EulerMethod implements AbstractClass{

    public EulerMethod(){}

    //Series of method
    XYSeries series1;
    //Series of method's error
    XYSeries series2;

    public void funct(Float step, Float x, Float y, Float xf){
        //Calculating points that satisfy to Euler's Method and adding them to the series1
        //and points that satisfy to Euler's Method error and adding them to the series2:
        series1 = new XYSeries( key: "Euler's Method");
        series1.add(x, y);
        series2 = new XYSeries( key: "Euler's Method Error");
        series2.add(x, y);
        float e = 2.71828f;
        float sum = 0f;
        for(float i = x; i <= xf; i += step){
            series1.add(i, y: y + step*my_function(y,i));

            double k = y + step*my_function(y,i);
            System.out.println(i + "  " + k);
            series2.add(i, y: y + step*my_function(y,i) - Math.pow((2*Math.pow(e,i)-1),2));
            sum +=Math.abs( y + step*my_function(y,i) - Math.pow((2*Math.pow(e,i)-1),2));
            double f = step*my_function(y,i);
            y += (float)f;
        }
        sum/= ((xf-x)/step);
        System.out.println(sum);
    }
    public float my_function(double y,float x){
        float e = 2.71828f;
        double f = 2*y+2*Math.sqrt(y);
        return (float)f;
    }
}
```

3. Improved Euler

```
    for(float i = x; i <= xf; i += step){
        series1.add(i, y: y + step*my_function(y,i));

        double k = y + step*my_function(y,i);
        System.out.println(i + "   " + k);
        series2.add(i, y: y + step*my_function(y,i) - Math.pow((2*Math.pow(e,i)-1),2));
        sum +=Math.abs( y + step*my_function(y,i) - Math.pow((2*Math.pow(e,i)-1),2));
        double f = step*my_function(y,i);
        y += (float)f;
    }
    sum/= ((xf-x)/step);
    System.out.println(sum);
}
public float my_function(double y,float x){
    float e = 2.71828f;
    double f = 2*y+2*Math.sqrt(y);
    return (float)f;
}
```

4. Runge-Kutta method

```
        float k1, k2, k3, k4;
        float e = 2.71828f;
        for(float i = x+step; i < xf; i += step){
            k1 = my_function(y,x);
            k2 = my_function( y: y+step*k1/2,  x: x+step/2);
            k3 = my_function( y: y+step*k2/2,  x: x+step/2);
            k4 = my_function( y: y+step*k2,  x: x+step);
            series1.add(i,  y: y+step/6*(k1+2*k2+2*k3+k4));
            series2.add(i,  y: y+step/6*(k1+2*k2+2*k3+k4) -  Math.pow((2*Math.pow(e,i)-1),2));
            y += step/6*(k1+2*k2+2*k3+k4);
            if (y>30)
                break;
        }
    }
    public float my_function(float y,float x){
        float e = 2.71828f;
        double f = 2*y+2*Math.sqrt(y);
        return (float)f;
    }
}
```