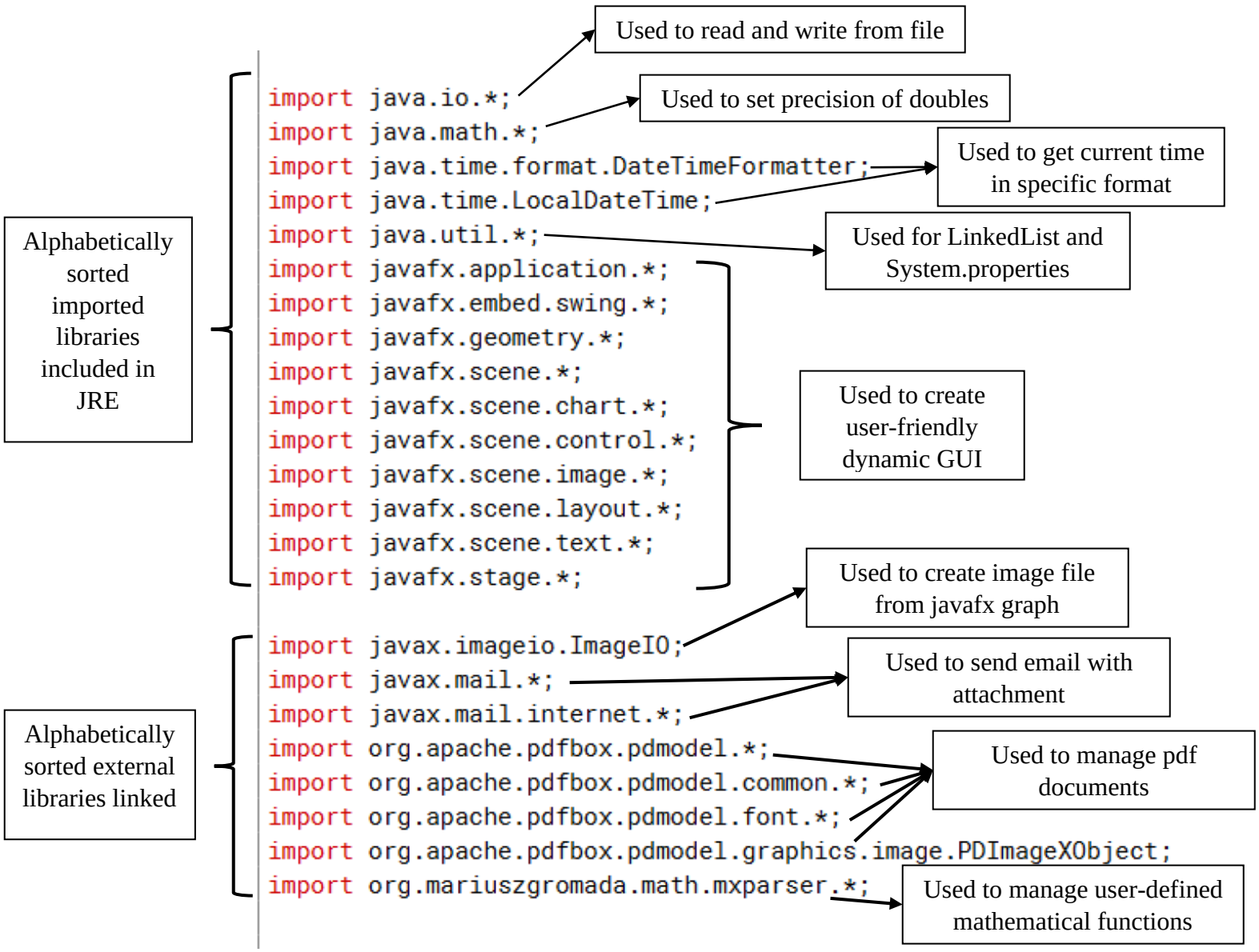
Development

Contents

1. Libraries used – Abstraction

Imported libraries play a major role in the program. They offer the chance to complete complicated tasks with simple commands, such as simple statements, function calls and variable definitions. They make coding easier and less time-consuming, since pre-existing code does not need to be written from scratch, as well as less error-prone and easier to debug, as the lines of code included in libraries have been tested by millions of users and achieve their aim beyond doubt. Finally, when using pre-compiled code, we do not need to know **how** each function works; abstraction exists.

Figure 1.1: Libraries imported in main Class



It is worth noting how the external libraries are used. javax.mail[i] and Apache PDFBox[ii] are presented below, while mXparser[iii] (org.mariuszgromada) will be presented throughout the document.

Figure 1.2: Using javax.mail to send email[iv]

```
//send pdf created to students of currentClass
public void email()
{
    if(curFileName.equals(""))
        return;
    File file = new File(curFileName);
    final String userEmail = "vissarionchristodoulou@gmail.com";
    final String userPassword = "vissArion100%";
    Properties p = System.getProperties();
    p.setProperty("mail.smtp.host", "smtp.gmail.com");
    p.put("mail.smtp.auth", "true");
    p.put("mail.smtp.starttls.enable", "true");
    Session s = Session.getDefaultInstance(p,
                                           new javax.mail.Authenticator()
                                           {
                                               protected PasswordAuthentication getPasswordAuthentication()
                                               {
                                                   return new PasswordAuthentication(userEmail, userPassword);
                                               }
                                           });
    try
    {
        for(Class cur:classes)
        {
            if(cur.getName().equals(currentClass))
            {
                MimeMessage m = new MimeMessage(s);
                m.setFrom(new InternetAddress(userEmail));
                for(Student st:cur.getStudents())
                    m.addRecipient(Message.RecipientType.TO, new InternetAddress(st.getEmailAddress()));
                m.setSubject("Class diagrams");

                BodyPart mB = new MimeBodyPart();
                mB.setText("Automated email with class diagrams and corresponding data");
                MimeBodyPart mB2 = new MimeBodyPart();
                mB2.attachFile(file);

                Multipart mult = new MimeMultipart();
                mult.addBodyPart(mB);
                mult.addBodyPart(mB2);

                m.setContent(mult);
                Transport.send(m);
                break;
            }
        }
    }
    catch(Exception ex){}
}
```

If no file has been created terminate function

curFileName is a String member-variable

Load file

Start new session, log in

Current class is a string member variable

Linear search to find active class

Add every student as recipient

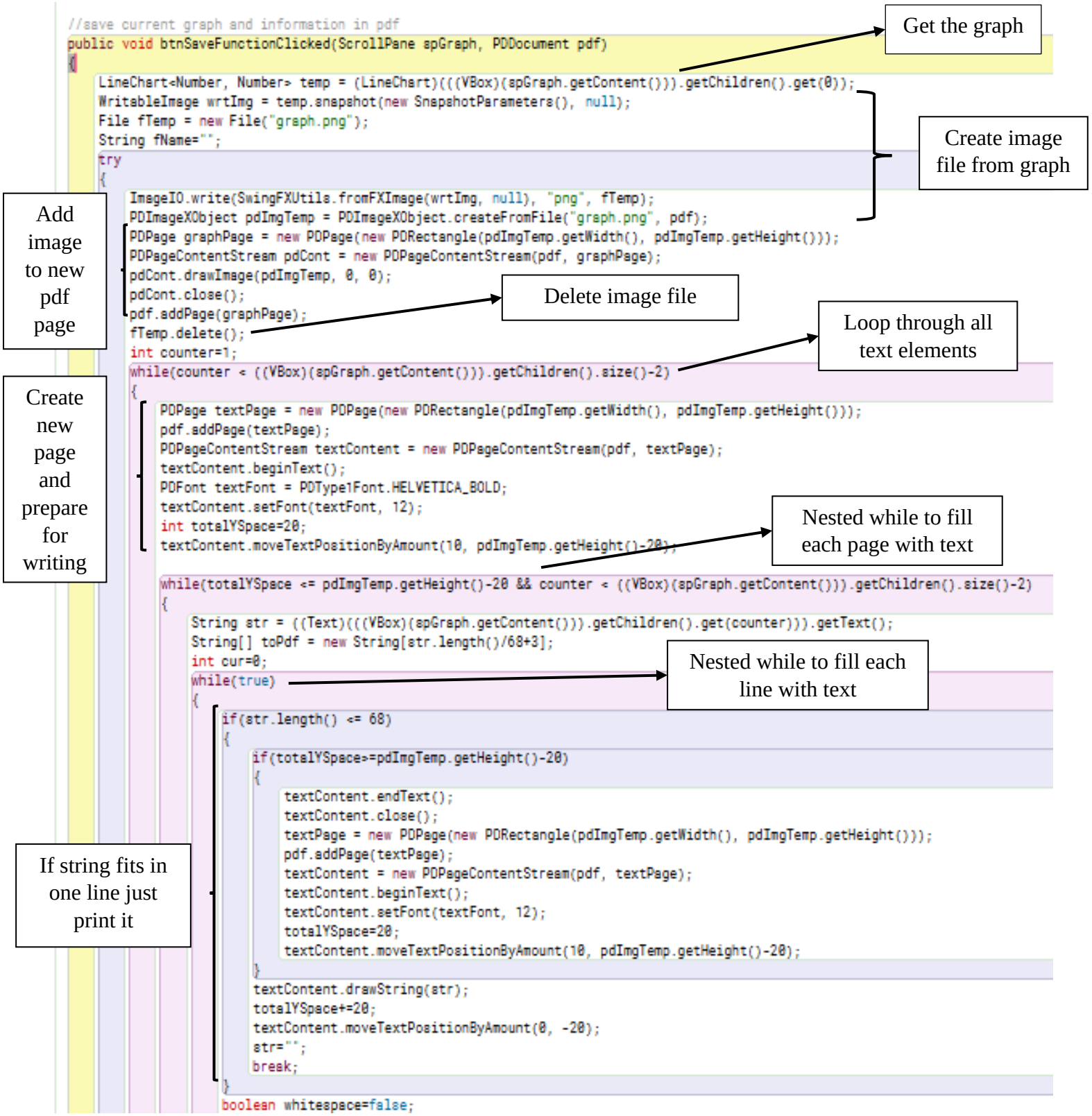
Set subject

Set text

Attach file

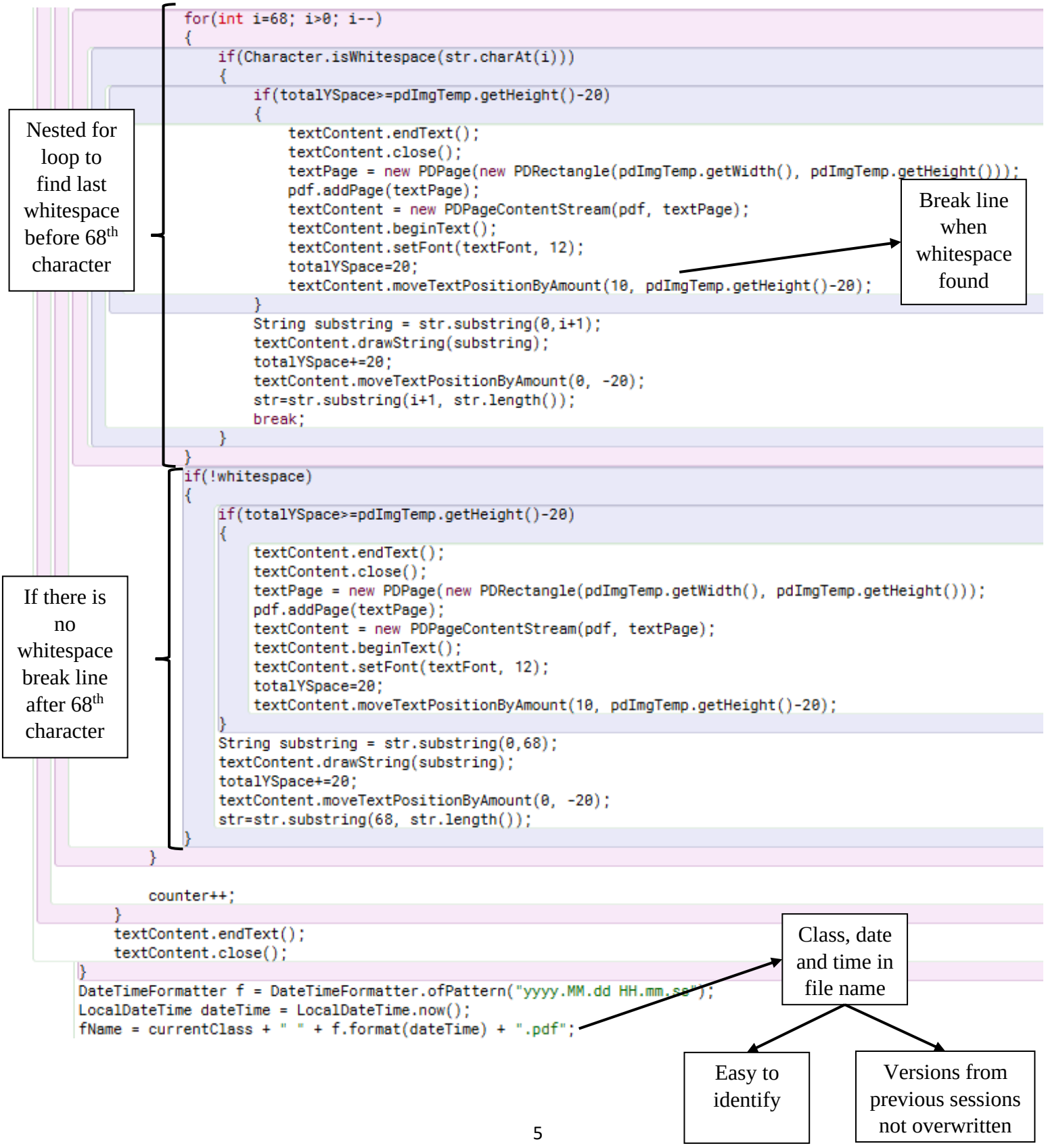
Send email

Figure 1.3a: Using Apache PDFBox to add to pdf




```
//save current graph and information in pdf
public void btnSaveFunctionClicked(ScrollPane spGraph, PDDocument pdf)
{
    LineChart<Number, Number> temp = (LineChart)(((VBox)(spGraph.getContent())).getChildren().get(0));
    WritableImage wrtImg = temp.snapshot(new SnapshotParameters(), null);
    File fTemp = new File("graph.png");
    String fName="";
    try
    {
        ImageIO.write(SwingFXUtils.fromFXImage(wrtImg, null), "png", fTemp);
        PDImageXObject pdImgTemp = PDImageXObject.createFromFile("graph.png", pdf);
        PDPage graphPage = new PDPage(new PDRectangle(pdImgTemp.getWidth(), pdImgTemp.getHeight()));
        PDPageContentStream pdCont = new PDPageContentStream(pdf, graphPage);
        pdCont.drawImage(pdImgTemp, 0, 0);
        pdCont.close();
        pdf.addPage(graphPage);
        fTemp.delete();
        int counter=1;
        while(counter < ((VBox)(spGraph.getContent())).getChildren().size()-2)
        {
            PDPage textPage = new PDPage(new PDRectangle(pdImgTemp.getWidth(), pdImgTemp.getHeight()));
            pdf.addPage(textPage);
            PDPageContentStream textContent = new PDPageContentStream(pdf, textPage);
            textContent.beginText();
            PDFont textFont = PDType1Font.HELVETICA_BOLD;
            textContent.setFont(textFont, 12);
            int totalYSpace=20;
            textContent.moveTextPositionByAmount(10, pdImgTemp.getHeight()-20);
            while(totalYSpace <= pdImgTemp.getHeight()-20 && counter < ((VBox)(spGraph.getContent())).getChildren().size()-2)
            {
                String str = ((Text)(((VBox)(spGraph.getContent())).getChildren().get(counter))).getText();
                String[] toPdf = new String[str.length()/68+3];
                int cur=0;
                while(true)
                {
                    if(str.length() <= 68)
                    {
                        if(totalYSpace>=pdImgTemp.getHeight()-20)
                        {
                            textContent.endText();
                            textContent.close();
                            textPage = new PDPage(new PDRectangle(pdImgTemp.getWidth(), pdImgTemp.getHeight()));
                            pdf.addPage(textPage);
                            textContent = new PDPageContentStream(pdf, textPage);
                            textContent.beginText();
                            textContent.setFont(textFont, 12);
                            totalYSpace=20;
                            textContent.moveTextPositionByAmount(10, pdImgTemp.getHeight()-20);
                        }
                        textContent.drawString(str);
                        totalYSpace+=20;
                        textContent.moveTextPositionByAmount(0, -20);
                        str="";
                        break;
                    }
                    boolean whitespace=false;
```

Figure 1.3b: Using Apache PDFBox to add to pdf (Continued)




```
for(int i=68; i>0; i--)
{
    if(Character.isWhitespace(str.charAt(i)))
    {
        if(totalYSpace>=pdImgTemp.getHeight()-20)
        {
            textContent.endText();
            textContent.close();
            textPage = new PDPage(new PDRectangle(pdImgTemp.getWidth(), pdImgTemp.getHeight()));
            pdf.addPage(textPage);
            textContent = new PDPageContentStream(pdf, textPage);
            textContent.beginText();
            textContent.setFont(textFont, 12);
            totalYSpace=20;
            textContent.moveTextPositionByAmount(10, pdImgTemp.getHeight()-20);
        }
        String substring = str.substring(0,i+1);
        textContent.drawString(substring);
        totalYSpace+=20;
        textContent.moveTextPositionByAmount(0, -20);
        str=str.substring(i+1, str.length());
        break;
    }
}
if(!whitespace)
{
    if(totalYSpace>=pdImgTemp.getHeight()-20)
    {
        textContent.endText();
        textContent.close();
        textPage = new PDPage(new PDRectangle(pdImgTemp.getWidth(), pdImgTemp.getHeight()));
        pdf.addPage(textPage);
        textContent = new PDPageContentStream(pdf, textPage);
        textContent.beginText();
        textContent.setFont(textFont, 12);
        totalYSpace=20;
        textContent.moveTextPositionByAmount(10, pdImgTemp.getHeight()-20);
    }
    String substring = str.substring(0,68);
    textContent.drawString(substring);
    totalYSpace+=20;
    textContent.moveTextPositionByAmount(0, -20);
    str=str.substring(68, str.length());
}
}

counter++;
}
textContent.endText();
textContent.close();
}
DateTimeFormatter f = DateTimeFormatter.ofPattern("yyyy.MM.dd HH.mm.ss");
LocalDateTime dateTime = LocalDateTime.now();
fName = currentClass + " " + f.format(dateTime) + ".pdf";
```

Figure 1.3c: Using Apache PDFBox to add to pdf (Continued)

```
        prevFileName = curFileName;
        if(!prevFileName.equals(""))
        {
            File del = new File(prevFileName);
            del.delete();
        }
        curFileName = fName;
        pdf.save(curFileName);
    }
    catch(Exception e){}
}
```

Delete previous version from the same session, since it was modified

2. Object-Oriented Programming (OOP) – Abstraction

OOP, evident from the class dependencies depicted in Figure 2.1, is yet another example of abstraction. Each class does not need to know how the other classes work; it can just access their public fields. Moreover, objects represent real world entities and through them, a more precise syntax can be acquired of the form subject.method(object) instead of method(subject, object) (in this context object refers to the syntactical position in an English sentence). A characteristic example of the use of classes is the use of the RangedFunction class in the code in Figure 2.2. Instead of using parallel arrays for the functions and their upper and lower bounds, I preferred to use objects of type RangedFunction that can be passed with less code from one method to another as arguments, while simultaneously the intrinsic relationship between the function and its boundaries is depicted.

Figure 2.1: Class Connections

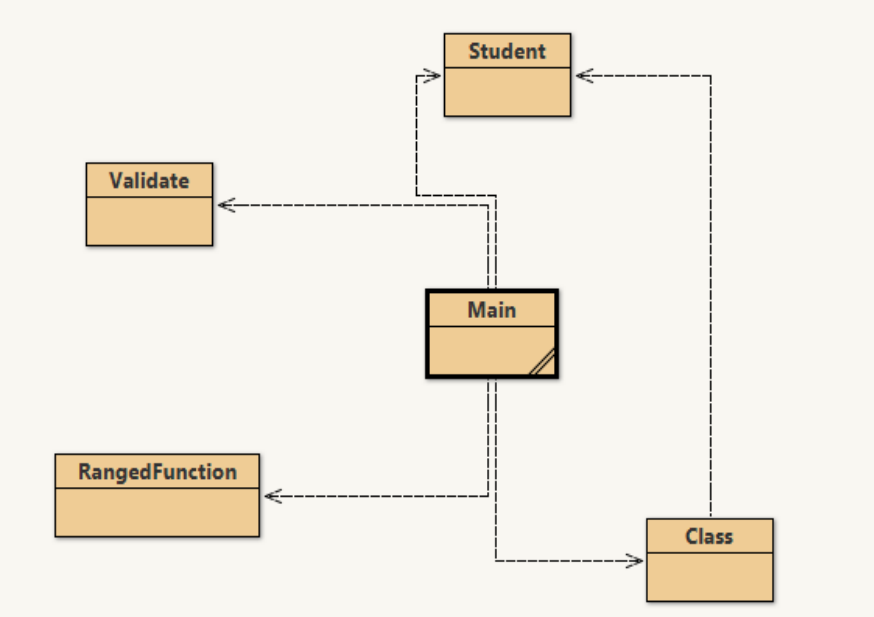


Figure 2.2a: Use of RangedFunction objects instead of parallel arrays and validating user input

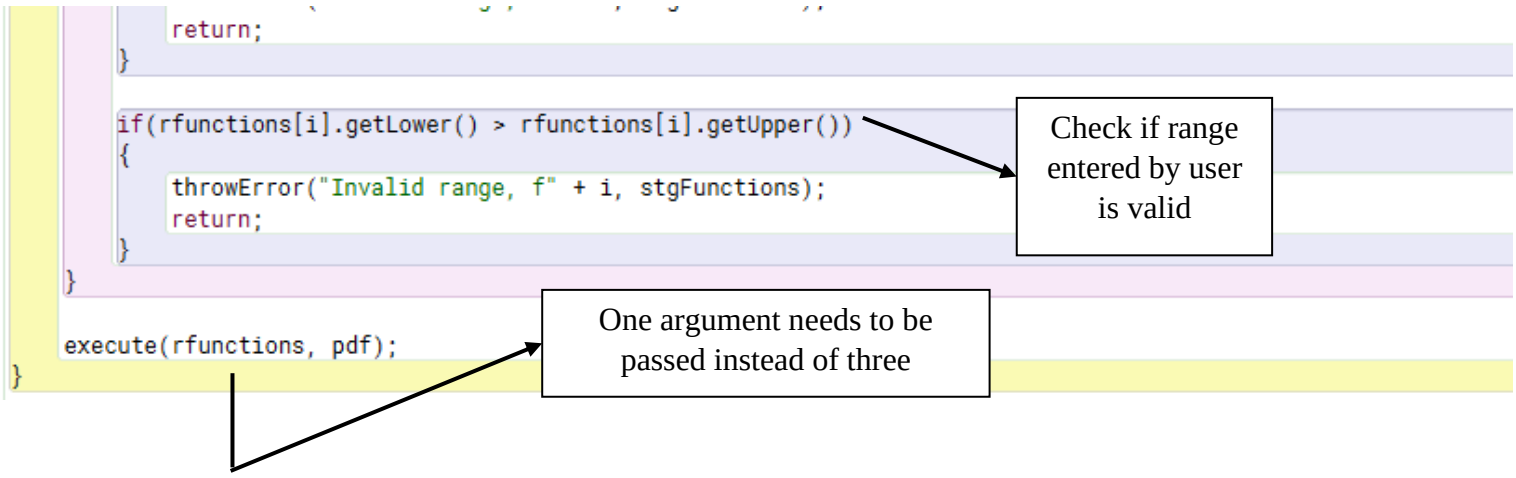
```
//Check the validity of the given functions before sketching the graph
public void btnExeClicked(VBox vbxFunctions, PDDocument pdf)
{
    if(vbxFunctions.getChildren().size() == 1)
    {
        throwError("No functions entered", stgFunctions);
        return;
    }

    RangedFunction[] rfunctions = new RangedFunction[vbxFunctions.getChildren().size()-1];
    for(int i=0; i<vbxFunctions.getChildren().size()-1; i++)
        rfunctions[i] = new RangedFunction();
    /*Function[] functions = new Function[vbxFunctions.getChildren().size()-1];
    double[] lower = new double[vbxFunctions.getChildren().size()-1];
    double[] upper = new double[vbxFunctions.getChildren().size()-1];*/
    for(int i=0; i<vbxFunctions.getChildren().size()-1; i++)
    {
        HBox hbxFunction = (HBox)vbxFunctions.getChildren().get(i);
        TextField txtfFunction = (TextField)hbxFunction.getChildren().get(1);
        String func = txtfFunction.getText();
        Function f = new Function("f" + i + "(x)=" + func);
        f.checkSyntax();
        if(f.getErrorMessage().substring(f.getErrorMessage().length()-19).equals("errors were found.\n"))
        {
            throwError("Invalid function syntax, f" + i, stgFunctions);
            return;
        }
        if(f.getFunctionExpressionString().equals(""))
        {
            throwError("Invalid function syntax, f" + i, stgFunctions);
            return;
        }

        rfunctions[i].setFunction(f);
        TextField txtfLower = (TextField)hbxFunction.getChildren().get(3);
        TextField txtfUpper = (TextField)hbxFunction.getChildren().get(5);
        String low = txtfLower.getText();
        String high = txtfUpper.getText();
        try
        {
            rfunctions[i].setLower(Double.parseDouble(low));
            rfunctions[i].setUpper(Double.parseDouble(high));
        }
        catch(Exception e)
        {
            throwError("Invalid range, f" + i, stgFunctions);
```

Instead of three arrays, function, lower and upper

Check if function entered by user is valid

Check if **numbers** were entered as range by user

Figure 2.2b: Use of RangedFunction objects instead of parallel arrays and validating user input (Continued)



3. Encapsulation – Abstraction

The use of encapsulation also serves as an example of abstraction. All member variables in all classes other than main are private and have corresponding accessor and mutator methods, as proved in Figures 3.1-3.3. This ensures that when a member variable is changed, it will be done through methods defined within the class and will, as a result, not be modified by mistake.

Figure 3.1a: Encapsulation and static polymorphism in class Student

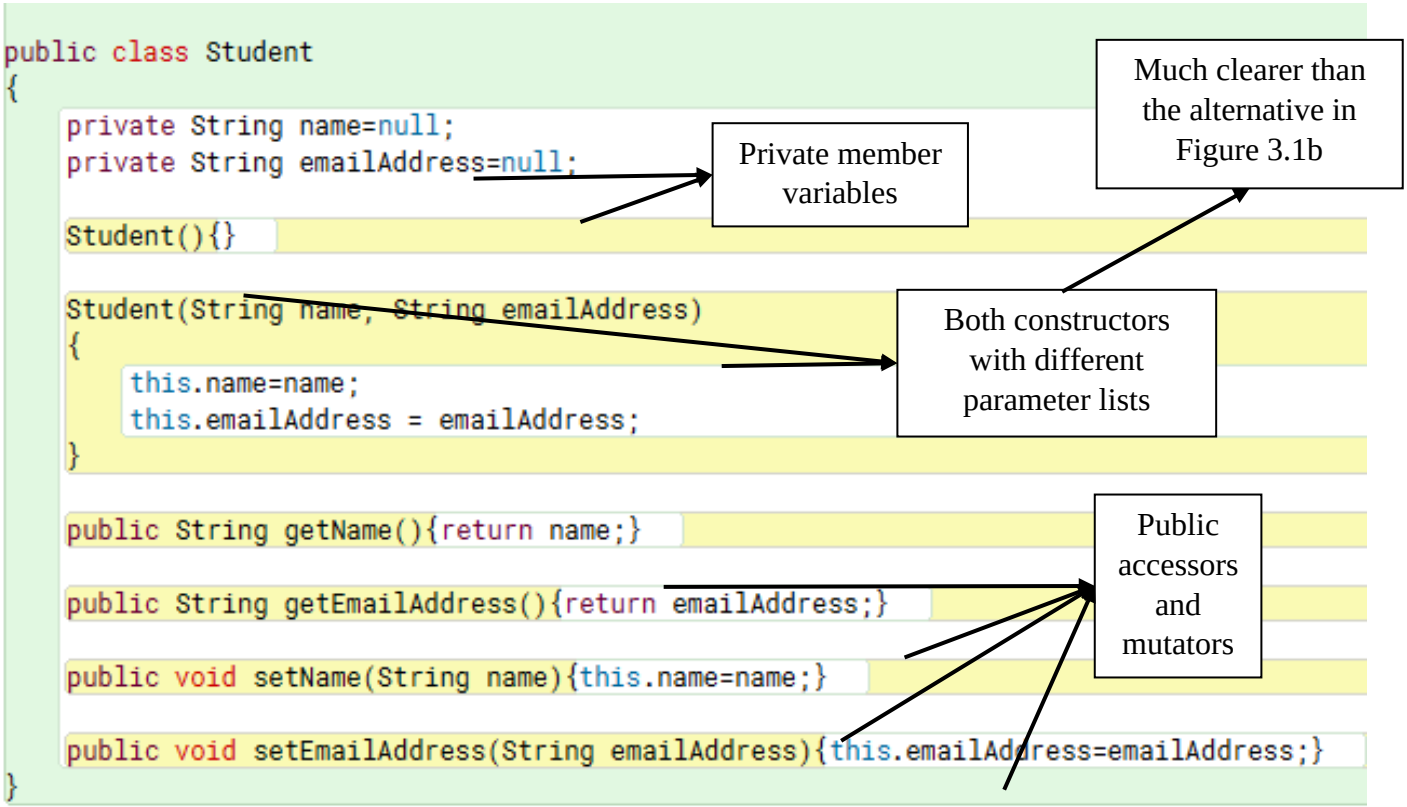


Figure 3.1b: Student without polymorphism

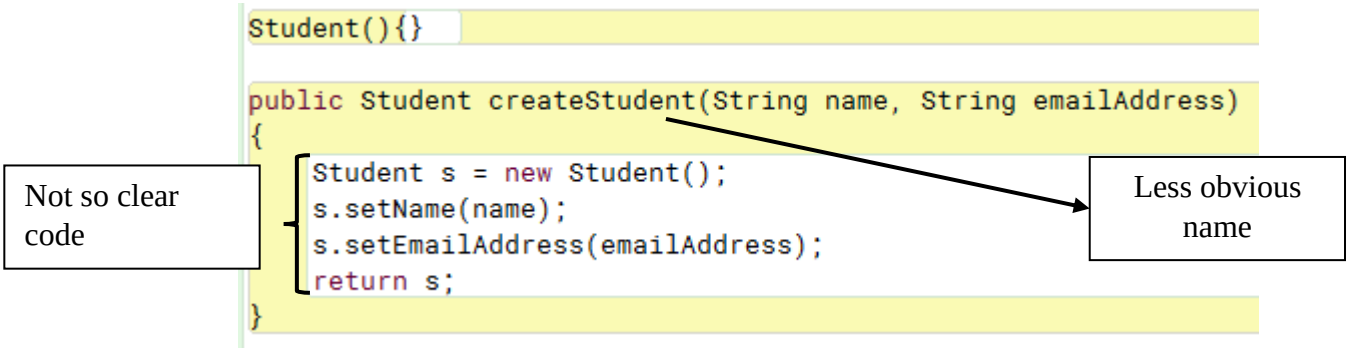


Figure 3.2: Encapsulation and static polymorphism in class Class

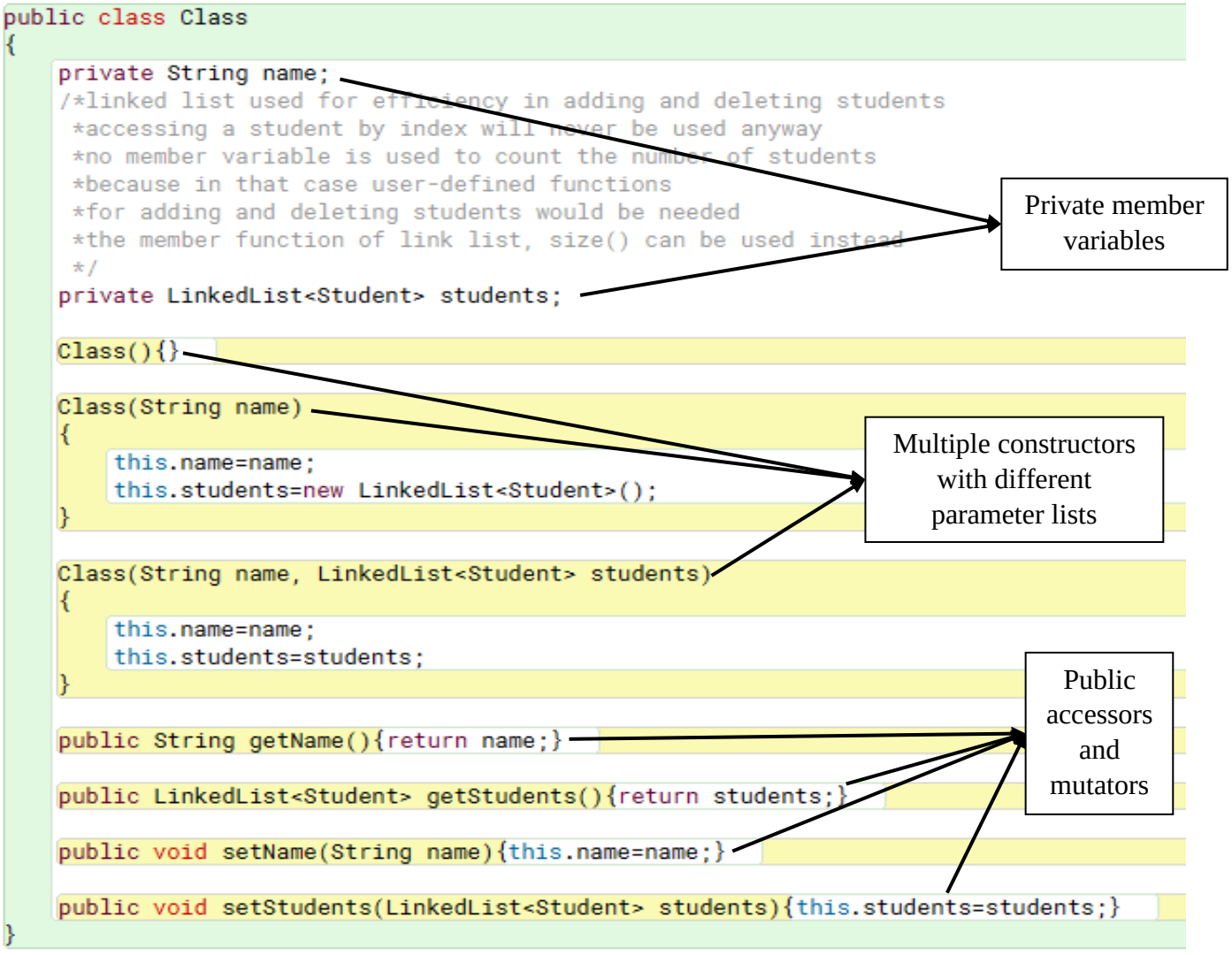
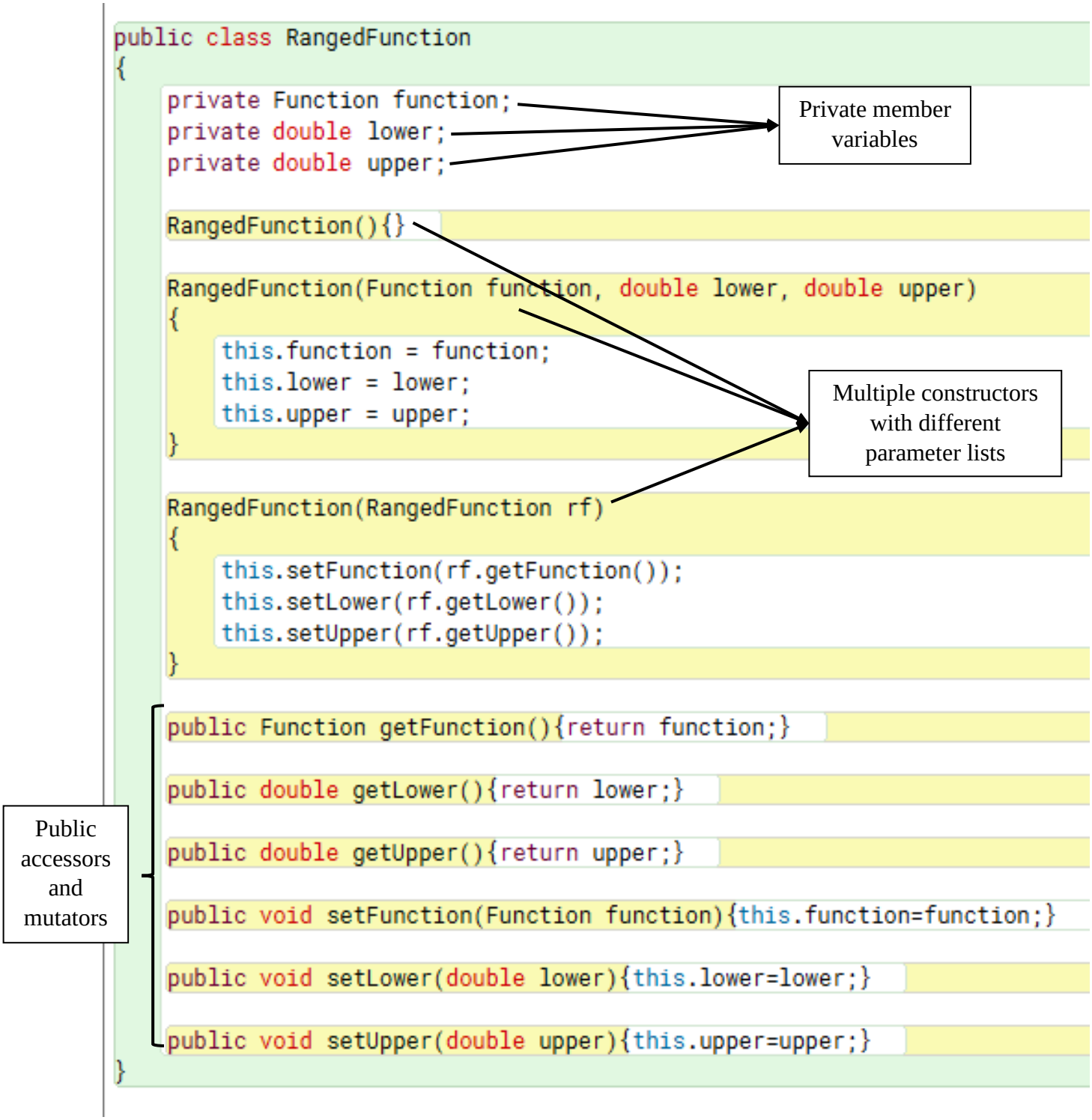


Figure 3.3: Encapsulation and static polymorphism in class RangedFunction




```
public class RangedFunction
{
    private Function function;
    private double lower;
    private double upper;

    RangedFunction(){}

    RangedFunction(Function function, double lower, double upper)
    {
        this.function = function;
        this.lower = lower;
        this.upper = upper;
    }

    RangedFunction(RangedFunction rf)
    {
        this.setFunction(rf.getFunction());
        this.setLower(rf.getLower());
        this.setUpper(rf.getUpper());
    }

    public Function getFunction(){return function;}

    public double getLower(){return lower;}

    public double getUpper(){return upper;}

    public void setFunction(Function function){this.function=function;}

    public void setLower(double lower){this.lower=lower;}

    public void setUpper(double upper){this.upper=upper;}
}
```

4. File Access

Except for working with pdfs, the program also interacts with a text file. It reads all information about Mr. Christos' classes and students when it starts running, and saves any changes when they take place. This provides a way to save information from one session to another, which is programmatically easy and efficient.

Figure 4.1: Reading classes from text file and try-catch blocks

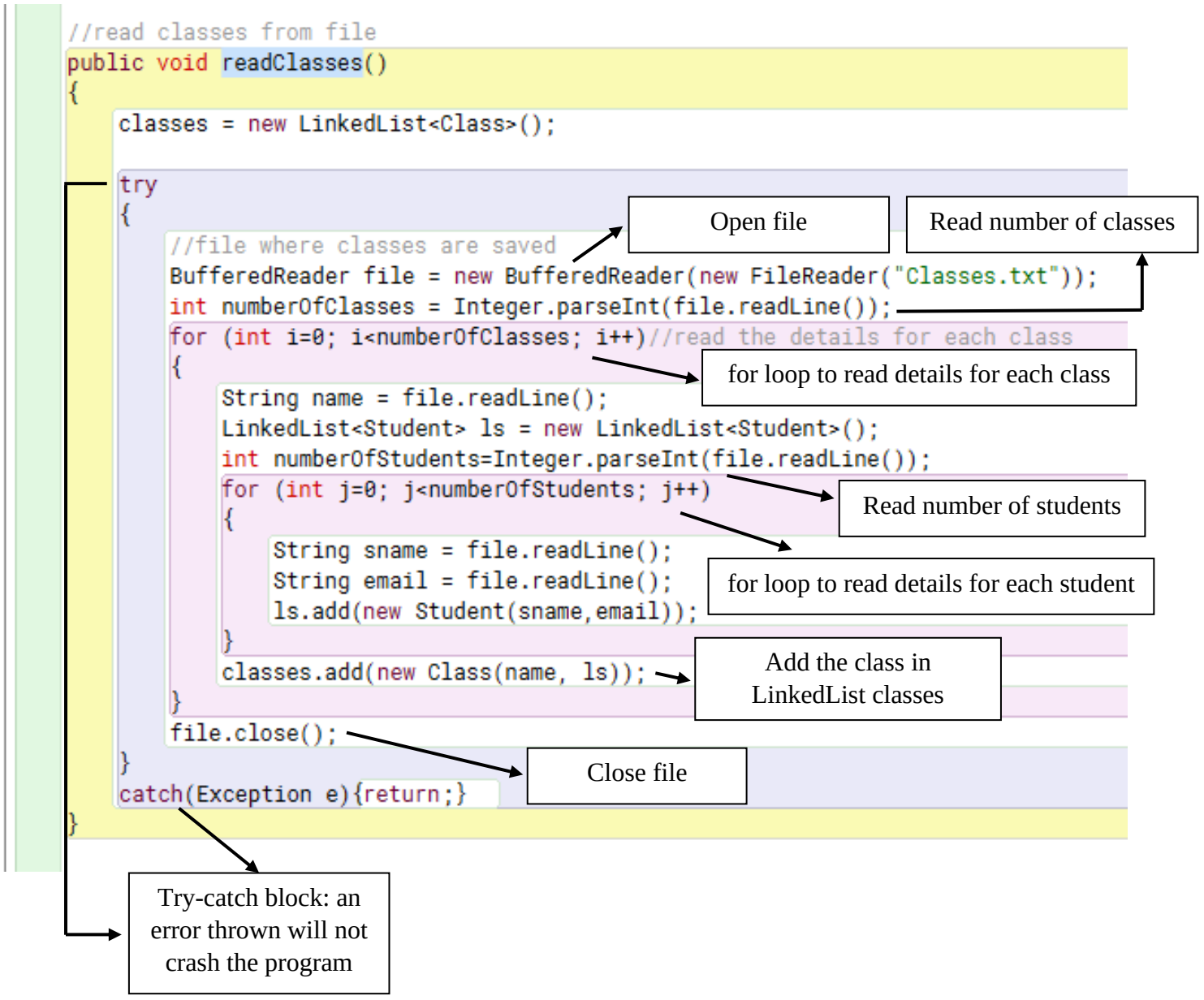


Figure 4.2: Saving classes to text file and try-catch blocks

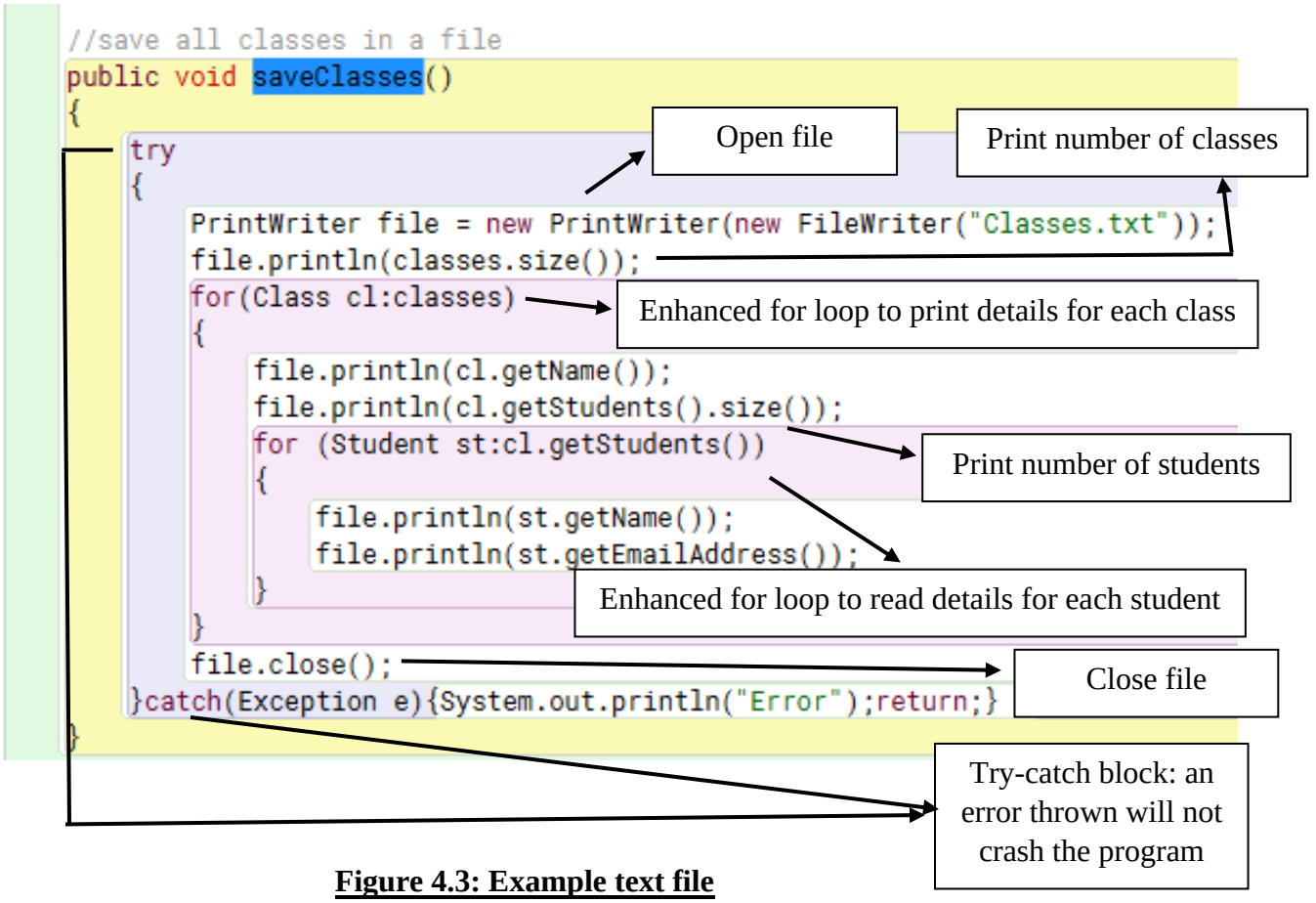
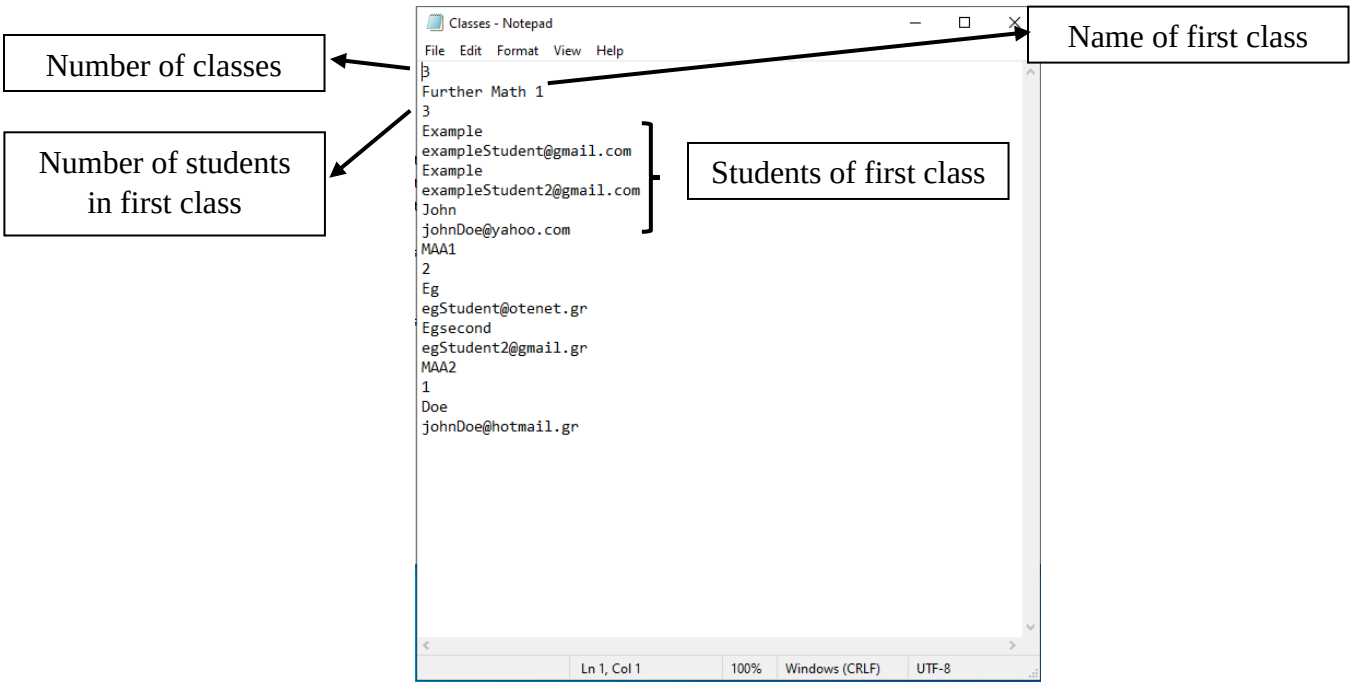


Figure 4.3: Example text file



5. Validation

Validation methods ensure that the user does not enter invalid data. It is essential to prevent the program from crashing and to avoid bugs. The Validate class, as shown in Figure 5.1, provides two methods for ensuring that when adding a new student, the name and email address are valid. As was noted in Figure 2.2, validation also takes place when the user inputs functions and ranges. Figure 5.3 outlines the behavior of the program when the user inputs invalid data in any field.

Figure 5.1: The Validate class

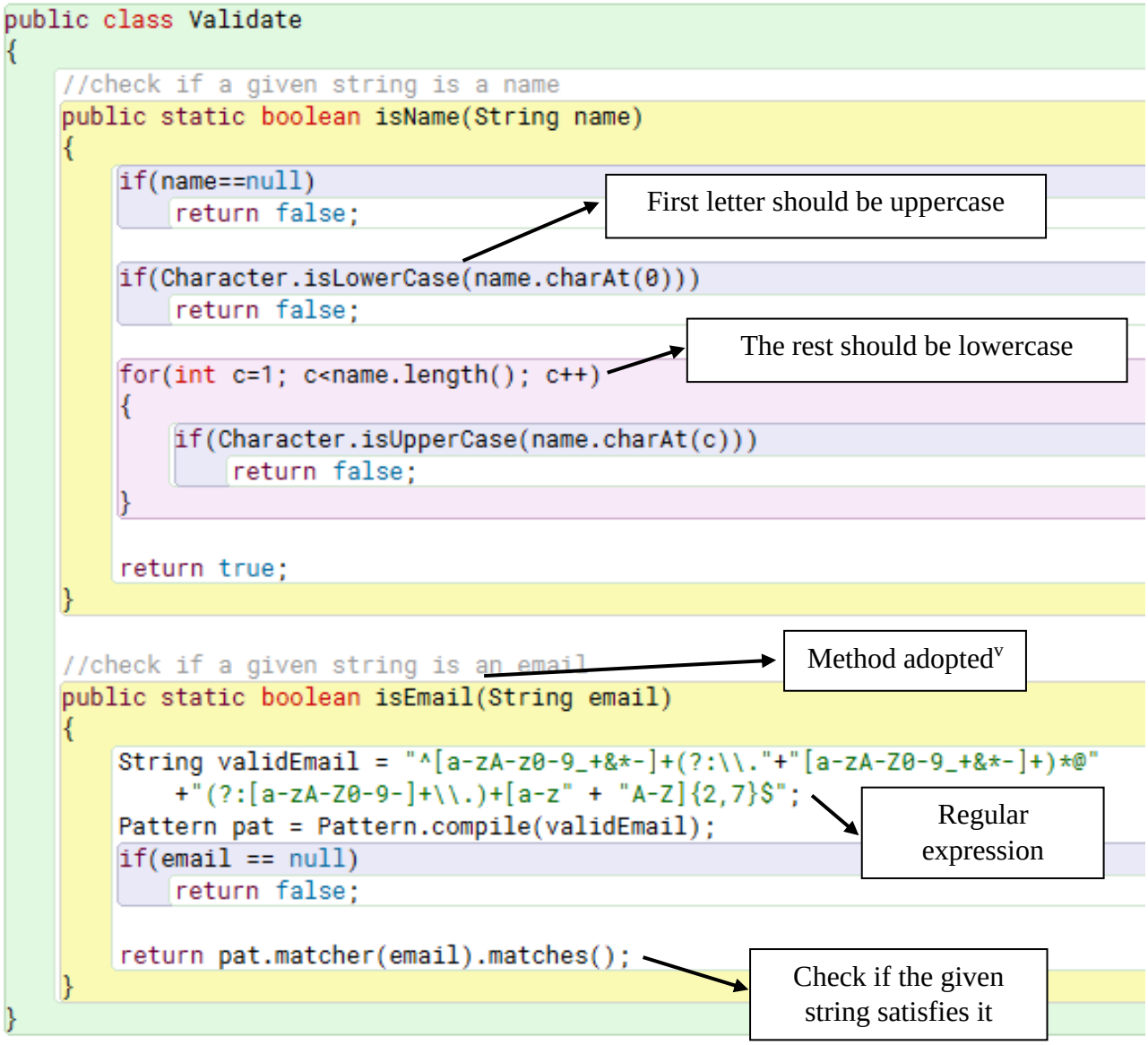


Figure 5.2: The validate class in use

```
//validate student name and email
public void btnDoneAddingStudentClicked(String clName, String name, String email)
{
    if (!(Validate.isName(name) && Validate.isEmail(email)))
    {
        throwError("Invalid name or email", stgAddStudent);
        return;
    }

    Class cl = new Class();
    for(Class cur:classes)
    {
        if(cur.getName().equals(clName))
        {
            cl=cur;
            break;
        }
    }

    for(Student st:cl.getStudents())
    {
        if(st.getEmailAddress().equals(email))
        {
            throwError("Email already exists", stgStudents);
            return;
        }
    }

    cl.getStudents().add(new Student(name, email));
    stgAddStudent.hide();
    btnStudentsClicked(clName);
}
```

Enhanced for loop: linear search

Enhanced for loop: uniqueness of email

Add student in class

Figure 5.3a: Entering invalid input

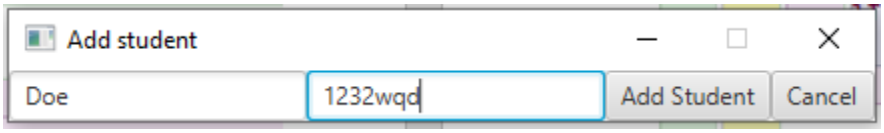
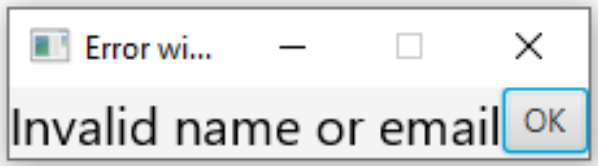


Figure 5.3b: The program's response to invalid input



6. Error handling

Error handling is also an essential part of this program, keeping it from crashing. It takes the form of try-catch blocks, and it may be a part of validating user input, as in Figure 2.2, or of ensuring that the program does not face unexpected errors, as explained in Figures 4.1 and 4.2.

7. Static polymorphism

Static polymorphism, achieved through function overloading, namely having multiple functions with the same names but different parameter lists, helps in creating programs where the method names are indeed representative of what the method does without being too descriptive. Moreover, they reduce code redundancy. These two points are explained in Figures 3.1-3.3.

8. Dynamic polymorphism

Dynamic polymorphism is what makes the javafx library usable in the first place, for it allows, by extending the Application class and overwriting its start method, to run the code on a JavaFX Application thread.

Figure 8.1: Overriding method start

```
/*this function runs in the beggining of the program
 *it displays the initial window
 *which contains a label, an image and two buttons
 *it also retrieves all the data that the program needs
 *and places it in variables available to the whole program
 */
@Override public void start(Stage dumpStg)
{
    readClasses();

    Text txtWelcome = new Text("Welcome!");
    txtWelcome.setFont(new Font(20));

    HBox hbxWelcome = new HBox();
    hbxWelcome.getChildren().add(txtWelcome);
    hbxWelcome.setAlignment(Pos.BASELINE_CENTER);

    Image imgMaths = new Image("Maths.jpg");
    ImageView imgvMaths = new ImageView(imgMaths);

    Button btnClasses = new Button("Classes");
    btnClasses.setMinWidth(166);
    btnClasses.setOnAction(e->btnClassesClicked());

    Button btnEmail = new Button("Email");
    btnEmail.setMinWidth(166);
    btnEmail.setOnAction(e->chooseClass(true));

    Button btnSession = new Button("New Session");
    btnSession.setMinWidth(166);
    btnSession.setOnAction(e->chooseClass(false));

    HBox hbxInitBtns = new HBox();
    hbxInitBtns.getChildren().addAll(btnClasses, btnEmail, btnSession);

    VBox vbxInit = new VBox();
    vbxInit.getChildren().addAll(hbxWelcome, imgvMaths, hbxInitBtns);

    stgInit = new Stage();
    Scene scnInit = new Scene(vbxInit);
    stgInit.setScene(scnInit);
    stgInit.setTitle("Welcome page");
    stgInit.resizableProperty().setValue(false);
    stgInit.show();
}
```

Override method start defined in superclass Application

Image copied[vi]

Event handlers →dynamic GUI

9. Graphs

One of the main purposes of the application created was to draw graphs. This is achieved by calculating the y-value for thousands of x-values within the function's domain, adding them as points in a set of axes and connecting these points with straight lines. The great number of points in a limited amount of pixels will give the graphs of the curves their real shape, meaning that the graph will look like a connection of straight lines, although essentially it is nothing more than that, as can be derived from Figure 9.1. The resulting graph and any data calculated on it can be seen in Figure 10.6.

Figure 9.1a: Drawing the graph

```
//plot the functions and allow any kind of calculations on them
public void execute(RangedFunction[] rfunctions, PDDocument pdf)
{
    stgFunctions.hide();

    VBox vbxGraph = new VBox();
    final NumberAxis xAxis = new NumberAxis();
    final NumberAxis yAxis = new NumberAxis();

    LineChart<Number, Number> graph = new LineChart<Number, Number>(xAxis, yAxis);
    graph.setCreateSymbols(false);

    XYChart.Series[] series = new XYChart.Series[rfunctions.length];
    for(int i=0; i<series.length; i++)
    {
        series[i] = new XYChart.Series();
        series[i].setName("y=f" + i + "=" + rfunctions[i].getFunction().getFunctionExpressionString());
        boolean num=false;
        for (double d=rfunctions[i].getLower(); d<=rfunctions[i].getUpper(); d+=(rfunctions[i].getUpper()-rfunctions[i].getLower())/2000)
        {
            double temp=rfunctions[i].getFunction().calculate(d);
            if (!Double.isNaN(temp))
            {
                if(!num)
                {
                    rfunctions[i].setLower(d);
                    num=true;
                }
                series[i].getData().add(new XYChart.Data(d, temp));
            }
            else if(num)
            {
                rfunctions[i].setUpper(((rfunctions[i].getUpper()-rfunctions[i].getLower())/2000));
            }
        }
        graph.getData().add(series[i]);
    }
```

Create graph with axis

for loop to add new curve for each function

Nested for loop to calculate the coordinates of 2000 successive points

Lower limit changed if function not defined for small values

Add point

Upper limit changed if function not defined for large values

Figure 9.1b: Drawing the graph (Continued)

```
    HBox hbxButtons1 = new HBox();
    HBox hbxButtons2 = new HBox();
    Button btnMax = new Button("Max");
    btnMax.setOnAction(e->chooseFunction(vbxGraph, rfunctions, "Max"));
    Button btnMin = new Button("Min");
    btnMin.setOnAction(e->chooseFunction(vbxGraph, rfunctions, "Min"));
    Button btnCalcY = new Button("Calculate y");
    btnCalcY.setOnAction(e->chooseFunction(vbxGraph, rfunctions, "Calculate y"));
    Button btnRoots = new Button("Roots");
    btnRoots.setOnAction(e->chooseFunction(vbxGraph, rfunctions, "Roots"));
    Button btnYIntercept = new Button("y-intercept");
    btnYIntercept.setOnAction(e->chooseFunction(vbxGraph, rfunctions, "YIntercept"));
    Button btnDerivative = new Button("Derivative");
    btnDerivative.setOnAction(e->chooseFunction(vbxGraph, rfunctions, "Derivative"));
    Button btnAreaUnder = new Button("Area under the graph");
    btnAreaUnder.setOnAction(e->chooseFunction(vbxGraph, rfunctions, "AreaUnder"));
    Button btnAreaBetweenX = new Button("Area between graph and x-axis");
    btnAreaBetweenX.setOnAction(e->chooseFunction(vbxGraph, rfunctions, "AreaBetween"));
    Button btnVolume = new Button("Volume of revolution");
    btnVolume.setOnAction(e->chooseFunction(vbxGraph, rfunctions, "Volume"));
    Button btnIntersection = new Button("Intersection");
    btnIntersection.setOnAction(e->chooseFunction(vbxGraph, rfunctions, "Intersection"));
    ScrollPane spGraph = new ScrollPane();
    Button btnSave = new Button("Save");
    btnSave.setOnAction(e->btnSaveFunctionClicked(spGraph, pdf));
    hbxButtons1.getChildren().addAll(btnMax, btnMin, btnCalcY, btnRoots, btnYIntercept, btnDerivative);
    hbxButtons2.getChildren().addAll(btnAreaUnder, btnAreaBetweenX, btnVolume, btnSave);

    if(rfunctions.length > 1)
        hbxButtons2.getChildren().add(3, btnIntersection);

    vbxGraph.getChildren().addAll(graph, hbxButtons1, hbxButtons2);
    spGraph.setContent(vbxGraph);
    spGraph.setMaxHeight(700);

    stgGraph = new Stage();
    stgGraph.resizableProperty().setValue(false);
    //if x button clicked or hotkey alt-f4 pressed return to initial screen
    stgGraph.setOnCloseRequest(e->{stgGraph.hide(); stgFunctions.show();});
    Scene scnGraph = new Scene(spGraph);
    stgGraph.setScene(scnGraph);
    stgGraph.setTitle("Graph");
    stgGraph.show();
}
```

Create buttons with event handlers

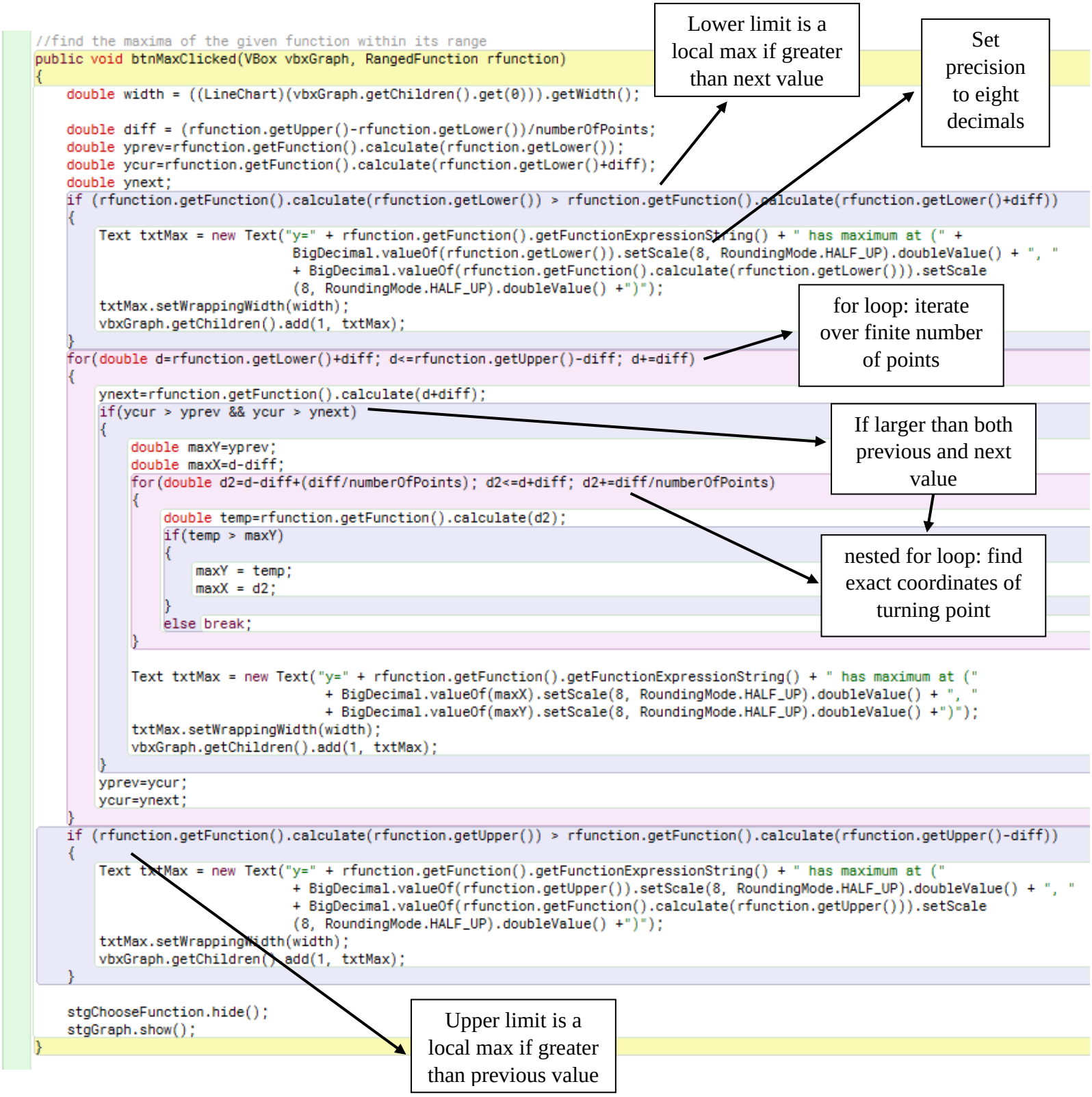
Add intersection button only if more than one functions were entered

10. Implementation of mathematical functions

The other crucial feature of the program was to be able to calculate certain values given a function. These have been implemented in a very brute-force way, presented in figures 10.1-10.5, which may lead to some error when an extremely large domain is given for x. Of course, I am referring to extreme cases, such as entering the function x^2 in the range (-10e5, 10e5), which are very rarely useful in real life, and even less so in a mathematics class. I opted for these algorithms because they terminate after a predetermined number of iterations, much faster than what it would need for a more sophisticated, recursive or iterative approach to converge. This, in combination with the limited precision that Mr. Christos requires, convinced me that the overhead of having to implement a more complicated solution was not worth it for this aspect of the program.

Maxima

Figure 10.1: Finding local maxima




```
//find the maxima of the given function within its range
public void btnMaxClicked(VBox vbxGraph, RangedFunction rfunction)
{
    double width = ((LineChart)(vbxGraph.getChildren().get(0))).getWidth();

    double diff = (rfunction.getUpper()-rfunction.getLower())/numberOfPoints;
    double yprev=rfunction.getFunction().calculate(rfunction.getLower());
    double ycur=rfunction.getFunction().calculate(rfunction.getLower()+diff);
    double ynext;
    if (rfunction.getFunction().calculate(rfunction.getLower()) > rfunction.getFunction().calculate(rfunction.getLower()+diff))
    {
        Text txtMax = new Text("y=" + rfunction.getFunction().getFunctionExpressionString() + " has maximum at (" +
                               BigDecimal.valueOf(rfunction.getLower()).setScale(8, RoundingMode.HALF_UP).doubleValue() + ", "
                               + BigDecimal.valueOf(rfunction.getFunction().calculate(rfunction.getLower())).setScale
                               (8, RoundingMode.HALF_UP).doubleValue() +")");
        txtMax.setWrappingWidth(width);
        vbxGraph.getChildren().add(1, txtMax);
    }
    for(double d=rfunction.getLower()+diff; d<=rfunction.getUpper()-diff; d+=diff)
    {
        ynext=rfunction.getFunction().calculate(d+diff);
        if(ycur > yprev && ycur > ynext)
        {
            double maxY=yprev;
            double maxX=d-diff;
            for(double d2=d-diff+(diff/numberOfPoints); d2<=d+diff; d2+=diff/numberOfPoints)
            {
                double temp=rfunction.getFunction().calculate(d2);
                if(temp > maxY)
                {
                    maxY = temp;
                    maxX = d2;
                }
                else break;
            }

            Text txtMax = new Text("y=" + rfunction.getFunction().getFunctionExpressionString() + " has maximum at ("
                                   + BigDecimal.valueOf(maxX).setScale(8, RoundingMode.HALF_UP).doubleValue() + ", "
                                   + BigDecimal.valueOf(maxY).setScale(8, RoundingMode.HALF_UP).doubleValue() +")");
            txtMax.setWrappingWidth(width);
            vbxGraph.getChildren().add(1, txtMax);
        }
        yprev=ycur;
        ycur=ynext;
    }
    if (rfunction.getFunction().calculate(rfunction.getUpper()) > rfunction.getFunction().calculate(rfunction.getUpper()-diff))
    {
        Text txtMax = new Text("y=" + rfunction.getFunction().getFunctionExpressionString() + " has maximum at ("
                               + BigDecimal.valueOf(rfunction.getUpper()).setScale(8, RoundingMode.HALF_UP).doubleValue() + ", "
                               + BigDecimal.valueOf(rfunction.getFunction().calculate(rfunction.getUpper())).setScale
                               (8, RoundingMode.HALF_UP).doubleValue() +")");
        txtMax.setWrappingWidth(width);
        vbxGraph.getChildren().add(1, txtMax);
    }

    stgChooseFunction.hide();
    stgGraph.show();
}
```

Minima

The code is similar.

y at x_0

Calculating y given an x-value x_0 is as easy as calling $f(x_0)$ after checking that lower bound < x_0 < upper bound

y-intercept

Simply call f(0) if lower bound < 0 < upper bound

Roots

Figure 10.2a: Finding roots

```
//find the roots of a function given within its range
public void btnRootsClicked(VBox vbxGraph, RangedFunction rfunction)
{
    double width = ((LineChart)(vbxGraph.getChildren().get(0))).getWidth();

    boolean rootFound=false;
    double diff=(rfunction.getUpper()-rfunction.getLower())/numberOfPoints;
    double yprev=rfunction.getFunction().calculate(rfunction.getLower());
    for(double d=rfunction.getLower()+diff; d<=rfunction.getUpper(); d+=diff)
    {
        double y=rfunction.getFunction().calculate(d);
        if((yprev<0 && y>0) || (yprev>0 && y<0))
        {
            rootFound=true;
            double closestToZero=Math.abs(yprev);
            double x=d-diff;
            for(double d2=d-diff; d2<=d; d2+=diff/numberOfPoints)
            {
                double temp=rfunction.getFunction().calculate(d2);
                if(Math.abs(temp)<closestToZero)
                {
                    closestToZero = Math.abs(temp);
                    x=d2;
                }
            }
            Text txtRoot = new Text("y=" + rfunction.getFunction().getFunctionExpressionString() + " has root at x="
                                    + BigDecimal.valueOf(x).setScale(8, RoundingMode.HALF_UP).doubleValue());
            txtRoot.setWrappingWidth(width);
            vbxGraph.getChildren().add(1, txtRoot);
        }
        yprev=y;
    }

    yprev=rfunction.getFunction().calculate(rfunction.getLower());
    double ycur=rfunction.getFunction().calculate(rfunction.getLower()+diff);
    double ynext;
    for(double d=rfunction.getLower()+diff; d<=rfunction.getUpper()-diff; d+=diff)
    {
        ynext=rfunction.getFunction().calculate(d+diff);
        if(ycur > yprev && ycur > ynext && ynext<0 && yprev<0)
        {
            double closestY=-yprev;
            double closestX=d-diff;
            for(double d2=d-diff+(diff/numberOfPoints); d2<=d+diff; d2+=diff/numberOfPoints)
            {
                double temp=Math.abs(rfunction.getFunction().calculate(d2));
                if(temp < closestY)
                {
                    closestY = temp;
                    closestX = d2;
                }
                else break;
```

for loop: iterate over finite number of points

By Rolle's theorem, if this is true then a root exists

nested for loop: find exact coordinates of root

For better accuracy check this case which might find missed roots

nested for loop: find closest value to 0

Figure 10.2b: Finding roots (Continued)

```
            }
            if(closestY < 1e-5)
            {
                rootFound=true;
                Text txtRoot = new Text("y=" + rfunction.getFunction().getFunctionExpressionString() + " has root at x="
                                        + BigDecimal.valueOf(closestX).setScale(8, RoundingMode.HALF_UP).doubleValue());
                txtRoot.setWrappingWidth(width);
                vbxGraph.getChildren().add(1, txtRoot);
            }
        }
        yprev=ycur;
        ycur=ynext;
    }

    yprev=rfunction.getFunction().calculate(rfunction.getLower());
    ycur=rfunction.getFunction().calculate(rfunction.getLower()+diff);
    for(double d=rfunction.getLower()+diff; d<=rfunction.getUpper()-diff; d+=diff)
    {
        ynext=rfunction.getFunction().calculate(d+diff);
        if(ycur < yprev && ycur < ynext && ynext>0 && yprev>0)
        {
            double closestY=yprev;
            double closestX=d-diff;
            for(double d2=d-diff+(diff/numberOfPoints); d2<=d+diff; d2+=diff/numberOfPoints)
            {
                double temp=Math.abs(rfunction.getFunction().calculate(d2));
                if(temp < closestY)
                {
                    closestY = temp;
                    closestX = d2;
                }
                else break;
            }
            if(closestY < 1e-5)
            {
                rootFound=true;
                Text txtRoot = new Text("y=" + rfunction.getFunction().getFunctionExpressionString() + " has root at x="
                                        + BigDecimal.valueOf(closestX).setScale(8, RoundingMode.HALF_UP).doubleValue());
                txtRoot.setWrappingWidth(width);
                vbxGraph.getChildren().add(1, txtRoot);
            }
        }
        yprev=ycur;
        ycur=ynext;
    }
```

If close enough to 0, consider the point a root

For better accuracy check this case which might find missed roots

nested for loop: find closest value to 0

If close enough to 0, consider the point a root

Derivative at x_0

This is the rate of change at x_0. We can just see how f changes very close to x_0. The answer will thus be $\frac{f(x_0+h)-f(x_0-h)}{2h}$, where h is a small number.

Area under the graph

Figure 10.3: Finding Area Under Graph

```
//find the area under the graph of a given function and above the x-axis
public void btnAreaUnderClicked(VBox vbxGraph, RangedFunction rfunction)
{
    double width = ((LineChart)(vbxGraph.getChildren().get(0))).getWidth();

    double area=0;
    double diff=(rfunction.getUpper()-rfunction.getLower())/(numberOfPoints*5);
    double prev=rfunction.getFunction().calculate(rfunction.getLower());
    for(double d=rfunction.getLower()+diff; d<=rfunction.getUpper(); d+=diff)
    {
        double temp=rfunction.getFunction().calculate(d);
        area += (prev+temp)*diff/2;
        prev=temp;
    }
    Text txtAreaUnder = new Text("y=" + rfunction.getFunction().getFunctionExpressionString() + ", area under graph= " + area);
    txtAreaUnder.setWrappingWidth(width);
    vbxGraph.getChildren().add(1, txtAreaUnder);
    stgChooseFunction.hide();
    stgGraph.show();
}
```

Use trapezium method in f(x) for area under graph

Area between the graph and the x-axis

Figure 10.4: Finding Area Between Graph and x-axis

```
/*find the area between the graph of a given function and the x-axis
 *this is different from what the previous method does
 *as it only takes positive values
 */
public void btnAreaBetweenClicked(VBox vbxGraph, RangedFunction rfunction)
{
    double width = ((LineChart)(vbxGraph.getChildren().get(0))).getWidth();

    double area=0;
    double diff=(rfunction.getUpper()-rfunction.getLower())/(numberOfPoints*5);
    double prev=Math.abs(rfunction.getFunction().calculate(rfunction.getLower()));
    for(double d=rfunction.getLower()+diff; d<=rfunction.getUpper(); d+=diff)
    {
        double temp=Math.abs(rfunction.getFunction().calculate(d));
        area += (prev+temp)*diff/2;
        prev=temp;
    }
    Text txtAreaBetween = new Text("y=" + rfunction.getFunction().getFunctionExpressionString() + ", area between graph and x-axis= " + area);
    txtAreaBetween.setWrappingWidth(width);
    vbxGraph.getChildren().add(1, txtAreaBetween);
    stgChooseFunction.hide();
    stgGraph.show();
}
```

Use trapezium method in |f(x)| for area between graph and x-axis

Volume of revolution

Figure 10.5: Finding Volume of Revolution

```
//find the volume of revolution of the graph of a given function
public void btnVolumeClicked(VBox vbxGraph, RangedFunction rfunction)
{
    double width = ((LineChart)(vbxGraph.getChildren().get(0))).getWidth();

    double volume=0;
    double diff=(rfunction.getUpper()-rfunction.getLower())/(numberOfPoints*5);
    double prev=rfunction.getFunction().calculate(rfunction.getLower());
    for(double d=rfunction.getLower()+diff; d<=rfunction.getUpper(); d+=diff)
    {
        double temp=rfunction.getFunction().calculate(d);
        volume += Math.PI*(prev*prev+temp*temp)*diff/2;
        prev=temp;
    }
    Text txtVolume = new Text("y=" + rfunction.getFunction().getFunctionExpressionString() + ", volume of revolution= " + volume);
    txtVolume.setWrappingWidth(width);
    vbxGraph.getChildren().add(1, txtVolume);
    stgChooseFunction.hide();
    stgGraph.show();
}
```

Use trapezium method in $\pi f^2(x)$ for volume of revolution

Point of intersection of two functions

Assuming two functions f(x) and g(x), their points of intersection are the same as the roots of f(x)-g(x).

Figure 10.5a: Plotted function and data calculated

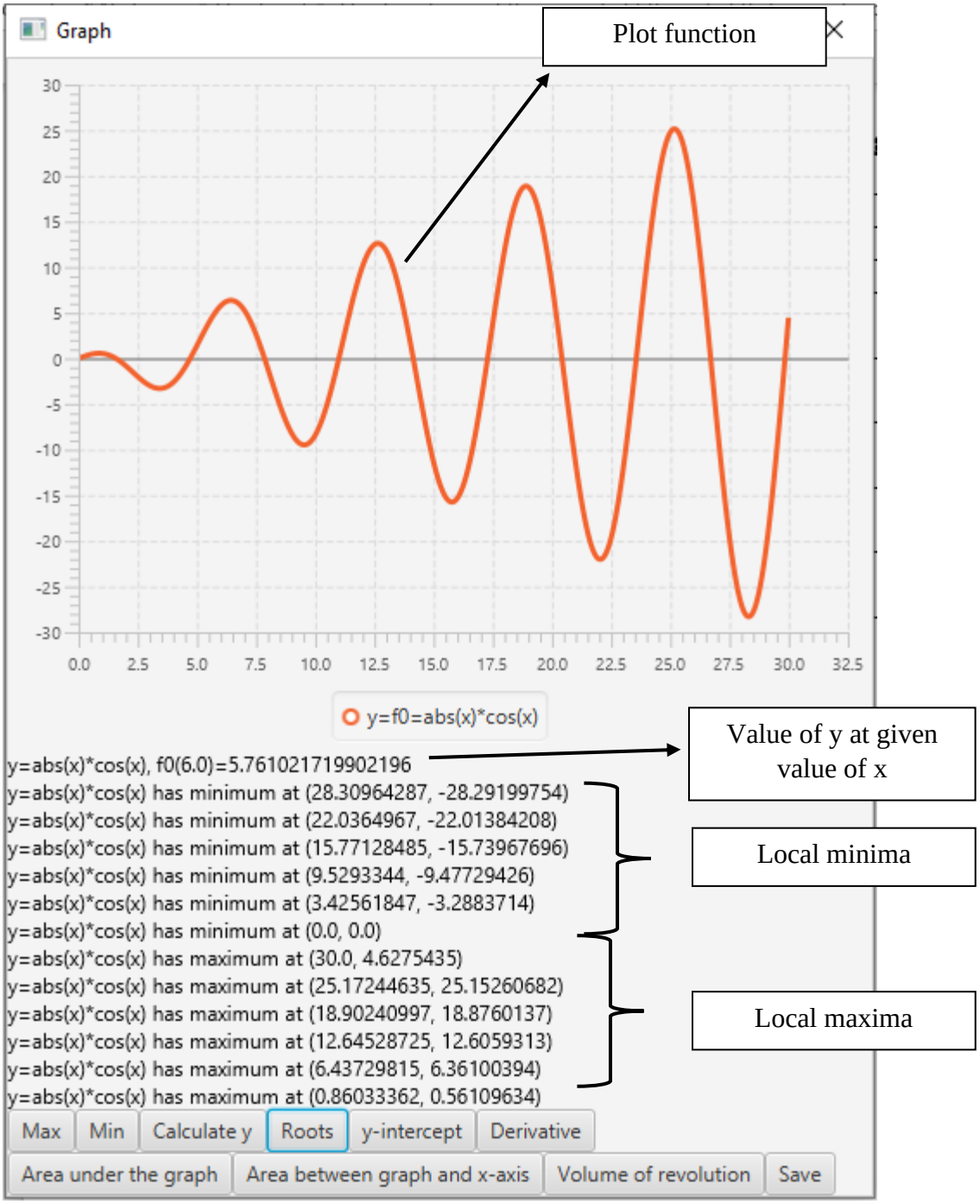


Figure 10.5b: Plotted function and data calculated (Continued)

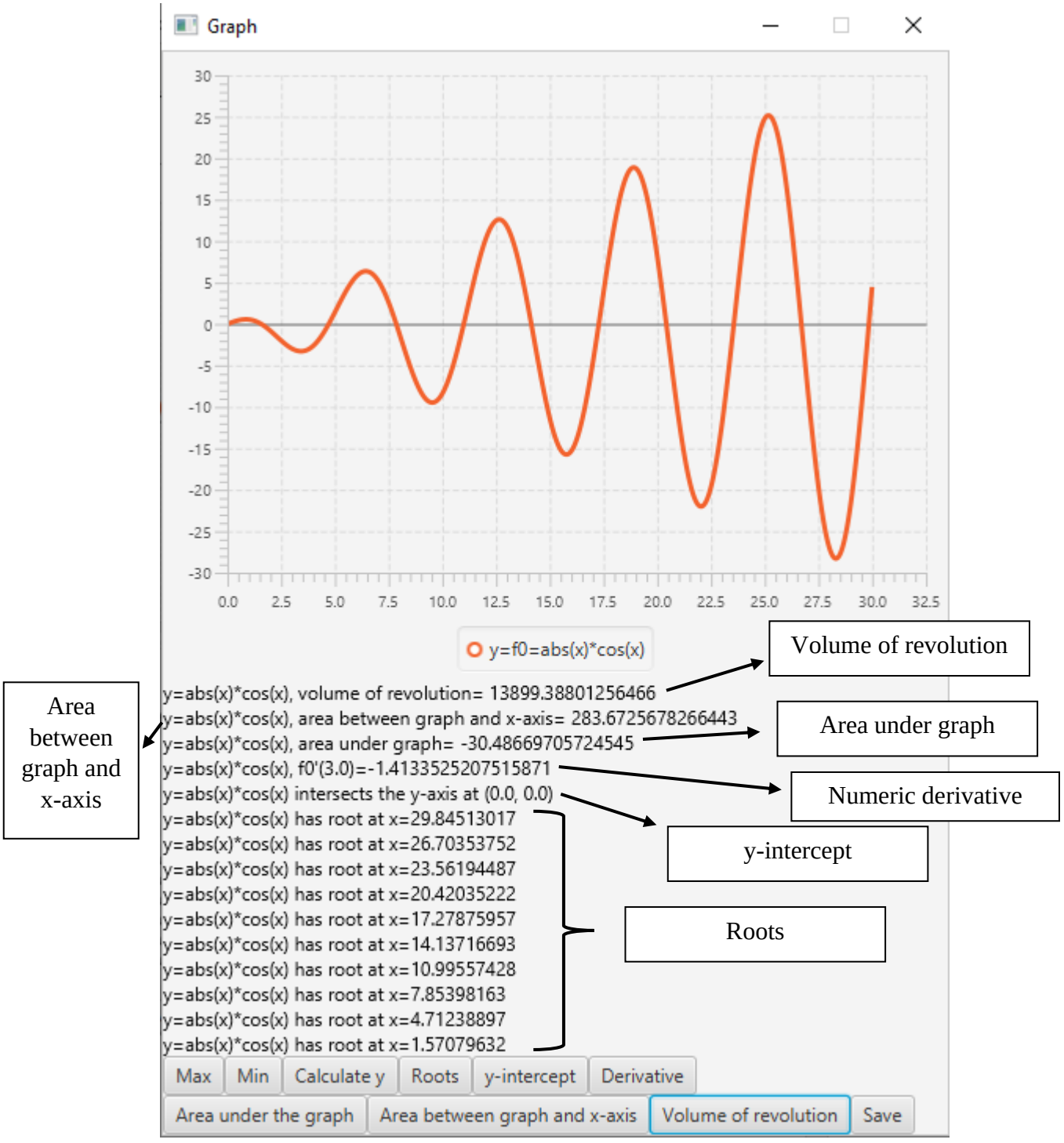
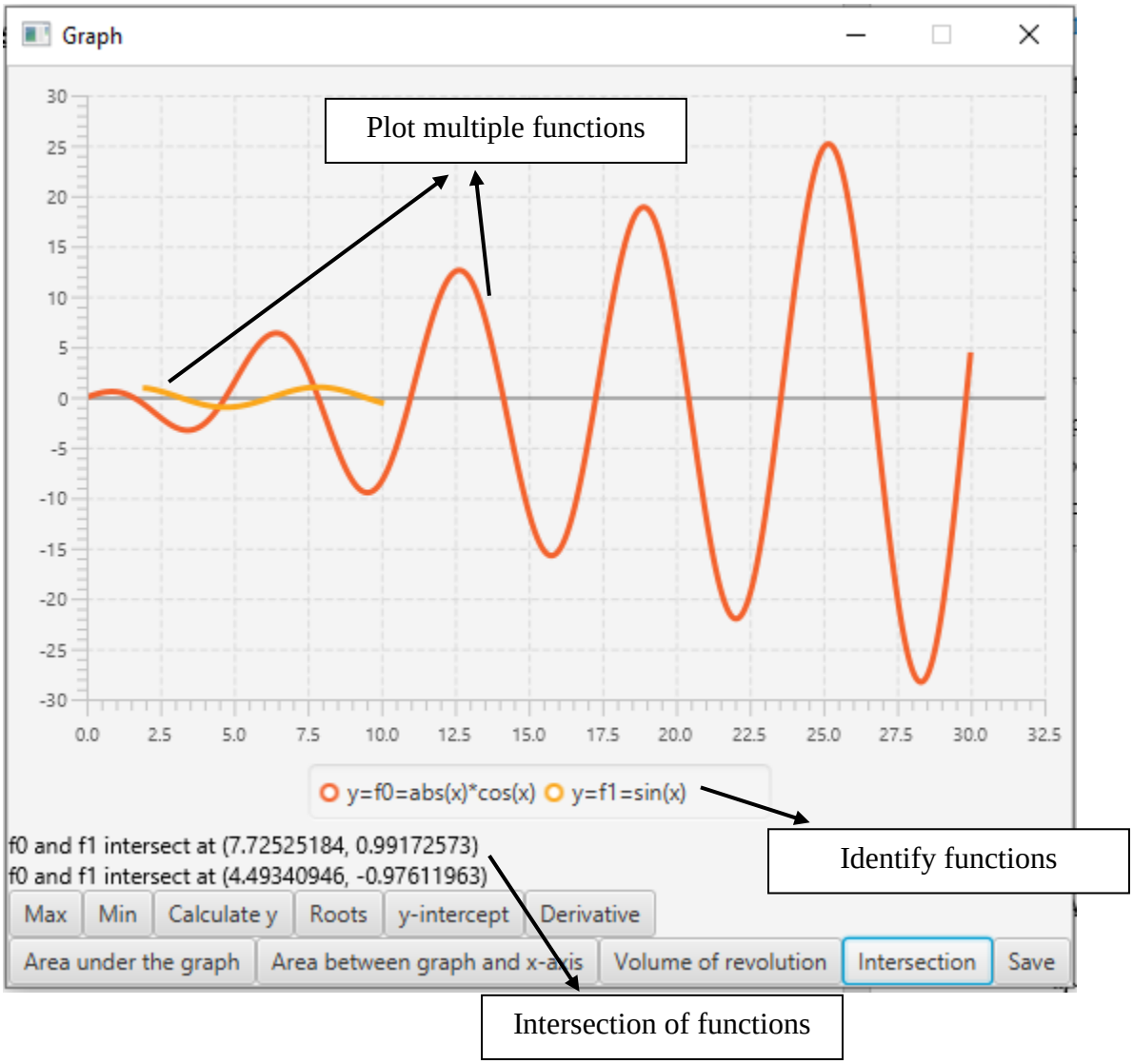


Figure 10.5c: Plotted function and data calculated (Continued)

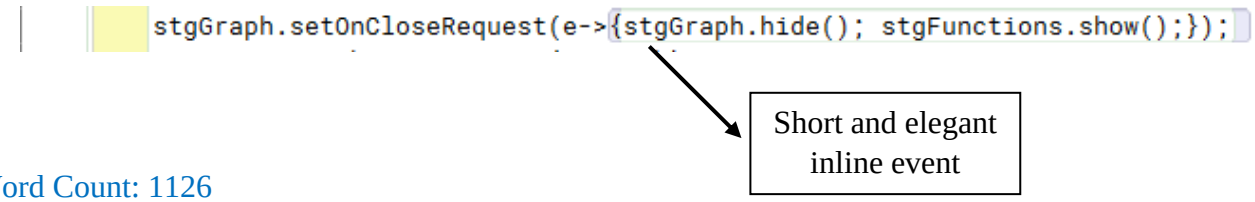


11. Inline events

Inline events, meaning blocks of code instead of method calls when an event-handler is called, can have a lot of advantages. Even though, under some circumstances they can do more harm than good to one's program, when used judicially, for very short methods that are not called many times, they can make both the code itself clearer and more readable and its execution slightly faster. The first is true because inline events reduce the vertical whitespace of the code, and also, when reading code with a lot of methods, one has to see where each one of them is

defined and keep track of where they were called, which is avoided by using inline events. As for faster execution, similarly to how a programmer might have difficulty reading a program with many methods, a compiler will have to stop and start execution from different points several times, while simultaneously the free space in the heap will be reduced, slowing down the execution, which can again be avoided by using inline events. Therefore, inline events have, to some degree, been exploited for the purposes of my program.

Figure 11.1: An example of an inline event



Word Count: 1126

References

"Apache PDFBox® - A Java PDF Library." Apache PDFBox | A Java PDF Library, pdfbox.apache.org/.

"Check If Email Address Valid or Not in Java." GeeksforGeeks, 4 Feb. 2018, www.geeksforgeeks.org/check-email-address-valid-not-java/.

"Complex Mathematics Images." Shutterstock, www.shutterstock.com/search/complex+mathematics.

Dimitriou, Kostas, and Markos Hatzitaskos. Core Computer Science: for the IB Diploma Program (International Baccalaureate). Express Publishing, 2018.

Dimitriou, Kostas, and Markos Hatzitaskos. Advanced Computer Science: for the IB Diploma Program (International Baccalaureate) High Level Computer Science. Express Publishing, 2018.

"Example of Sending Email in Java - Javatpoint." Www.javatpoint.com, www.javatpoint.com/example-of-sending-email-using-java-mail-api.

Java Platform Standard Edition 8 Documentation, docs.oracle.com/javase/8/docs/.

JavaMail, javaee.github.io/javamail/.

"MXparser – Math Expressions Parser for JAVA Android C# .NET/MONO/Xamarin – Mathematical Formula Parser / Evaluator Library." MXparser – Math Expressions Parser for JAVA Android C# .NET/MONO/Xamarin – Mathematical Formula Parser / Evaluator

Library, mathparser.org/.

[i] https://javaee.github.io/javamail/
[ii] https://pdfbox.apache.org/
[iii] http://mathparser.org/
[iv] Adopted code from https://www.javatpoint.com/example-of-sending-email-using-java-mail-api
[v] https://www.geeksforgeeks.org/check-email-address-valid-not-java/
[vi] https://www.shutterstock.com/search/complex+mathematics