Project 2

SSE 550

Dylan DeVries

Mercer University

October 29, 2021

Table of Contents

[1 Introduction 2](#_Toc86674111)

[2 User Interface 2](#_Toc86674112)

[Usage: 2](#_Toc86674113)

[Data File Structure: 3](#_Toc86674114)

[3 Covered Topics 5](#_Toc86674115)

[Data Accessing from File: 5](#_Toc86674116)

[Exception Handling: 5](#_Toc86674117)

[Code Organization with Packages: 6](#_Toc86674118)

[Subclassing: 7](#_Toc86674119)

[Abstract Class: 8](#_Toc86674120)

[4 Conclusion 8](#_Toc86674121)

[5 Appendix 9](#_Toc86674122)

[Proj2main.java 9](#_Toc86674123)

[ChartFrame.java 11](#_Toc86674124)

[ChartPanel.java 12](#_Toc86674125)

[AShape.java 13](#_Toc86674126)

[CBar.java 13](#_Toc86674127)

[CChart.java 14](#_Toc86674128)

[Cline.java 15](#_Toc86674129)

[CScatter.java 16](#_Toc86674130)

[IDrawable.java 17](#_Toc86674131)

# 

# 1 Introduction

For this project, students were required to demonstrate their knowledge of chapters seven through ten of the textbook. Covered topics in these chapters include subclassing, abstract classes, code organization with packages, proper exception handling, and data accessing from files. To implement these covered topics, I decided to use the java graphics class to make a program that will read in data from a specified file and then make line charts, scatter plots, and bar graphs from the data.

# 

# 2 User Interface

## Usage:

This program requires one user supplied command line argument to operate correctly. This argument is the name of the data file that will be read into the program. This data file will either need to be in the project folder, or the location needs to be specified within the command line argument. I have included a sample “data.txt” file within the project folder. Example usage is shown below.



If the user does not specify the command line argument the program will output an error message and close.



If the data file cannot be found, the program will output the error message shown below and close.



## Data File Structure:

There are 3 possible formats for data files, and each format will produce a different graph. The program will automatically generate a different type of graph based on how many numbers per line a data file contains. The first line of every datafile may contain an optional custom string title. If there is no string on the first line of the datafile, then the generated chart will have the default title.

The first possible format for a data file is one integer per line. This will produce a Bar Chart. The data points represent the height of the individual bars and will be plotted in the order in which they appear in the data file. Below is an example of a data file and its corresponding chart that will be generated.

Text, whiteboard

Description automatically generatedChart, bar chart

Description automatically generated

The second possible format for a data file is two integers per line. This will produce a scatter plot of the data. The first number on the line will represent the x-coordinate and the second number will represent the y-coordinate to form a data point. Below is an example of a scatter plot formed from a sample data file.

Text

Description automatically generatedChart

Description automatically generated with medium confidence

The last possible format for a data file is four integers per line. This will produce a line graph. The data line is arranged in the format of X1 Y1 X2 Y2, and a line will be generated between the two given points. Below is an example of a line graph formed from a data file.

Text

Description automatically generatedChart, line chart

Description automatically generated

# 3 Covered Topics

## Data Accessing from File:

All of the data that is used to create the graphs is read in from a data text file. To do this, I used a Scanner object and read in the file contents line by line. I then used the hasNextInt() function to grab the individual integers on the lines. Once I had all of the data from a given line, I called the appropriate function to store the data points, and then moved on to the next line until the entire file had been read.

Text

Description automatically generated

## Exception Handling:

When reading in data from a file, exceptions can occur. These exceptions have to be properly handled or it can cause an entire program to crash. To guard against this, I put the function call inside of a try/catch block of code. If the CreateChart() function throws an exception, the program will catch the exception and display the corresponding error on the console.

Text

Description automatically generated

To avoid other exceptions, I put all of my nextInt() function calls inside of hasNextInt() IF statements. This makes sure that the file line has a next int before attempting to assign it to a variable. This will guard against the NoSuchElementException and InputMismatchException.



## Code Organization with Packages:

The code for this project is separated into 3 separate packages: framework, gui, and proj2. This helps by segmenting similar code into sections that make navigating through the project easier. The framework package contains the java files necessary for making the bar charts, scatters plots, and line graphs. The gui package contains the java files necessary for actually creating the gui to display the graphs. Lastly the proj2 package is responsible for reading in the data from the file and calling the createChart() method.

A picture containing graphical user interface

Description automatically generated

## 

## Subclassing:

The AShape class in this project is an example of a Superclass. It contains two variables: refX and refY, which are used as reference points for plotting. The AShape class is then extended by the Cbar, CScatter, and CLine classes, which are the subclasses. All of these classes still have access to the reference variables in AShape, but extending the classes allows more specific functionality when plotting the different shapes.

Timeline

Description automatically generated with medium confidence

## Abstract Class:

The AShape class is an example of an abstract class. When this class is extended, it allows for the AShape() method to be defined in each of the subclasses. This allows for the subclasses to customize this method to be most suitable for them.

Text

Description automatically generated

# 4 Conclusion

In conclusion, I learned a lot from this project. Utilizing an Abstract class and Subclassing will help me in the future to become a better programmer. Also handling exceptions is a great skill to learn and was new to me as well. Utilizing the Java graphics class to make charts was a good application of these skills. I look forward to Project 3!

# 5 Appendix

## Proj2main.java

//Proj2Main.java

package proj2;

import java.awt.Color;

import java.io.\*;

import java.util.\*;

import framework.AShape;

import framework.CChart;

import framework.CLine;

import framework.CBar;

import framework.CScatter;

import gui.ChartFrame;

public class Proj2Main {

public static void main(String[] args) {

// Create your own chart by calling a createMyChart() method, and

// provide a title accordingly

//CChart aChart = createDummyChart();

if(args.length != 1){

System.out.println("Improper Usage: Please specify filename as command line arguement");

return;

}

String filename = args[0];

CChart aChart;

try{

aChart = createChart(filename);

}

catch(FileNotFoundException exc){

System.out.println("File Not Found");

System.out.println("");

return;

}

catch(NoSuchElementException exc){

System.out.println("No Such Element");

System.out.println("");

return;

}

new ChartFrame(aChart.title, aChart);

}

// Create your own chart by reading data from a file, handle exceptions

// properly, and think about line chart, scatter chart (with dots), and

// bar chart, etc.

private static CChart createChart(String filename) throws FileNotFoundException, NoSuchElementException{

int x1=0, x2=0, y1=0, y2=0;

Random randomNumbers = new Random();

String title = "";

int offset =0;

List<AShape> list = new ArrayList<AShape>();

Scanner dataFile = new Scanner(new File(filename));

while(dataFile.hasNextLine()){

String line = dataFile.nextLine();

Scanner scanner = new Scanner(line);

scanner.useDelimiter(" ");

if(scanner.hasNextInt() ){

x1 = scanner.nextInt();

if(scanner.hasNextInt() ){

y1 = scanner.nextInt();

if(scanner.hasNextInt() ){

x2 = scanner.nextInt();

if(scanner.hasNextInt() ){

y2 = scanner.nextInt();

Color color = new Color(randomNumbers.nextInt(256),

randomNumbers.nextInt(256),

randomNumbers.nextInt(256));

list.add(new CLine(x1, y1, x2, y2, color));

}

}

else{

Color color = new Color(randomNumbers.nextInt(256),

randomNumbers.nextInt(256),

randomNumbers.nextInt(256));

list.add(new CScatter(x1, y1, color));

}

}

else{

Color color = new Color(randomNumbers.nextInt(256),

randomNumbers.nextInt(256),

randomNumbers.nextInt(256));

list.add(new CBar(x1, offset, color));

offset += 28;

}

}

else{

if(scanner.hasNext()) title = scanner.next();

}

scanner.close();

}

AShape[] shapes = new AShape[list.size()];

list.toArray(shapes);

CChart ret = new CChart(shapes);

ret.title = title;

return ret;

}

}

## ChartFrame.java

//ChartFrame.java

package gui;

import javax.swing.JFrame;

import framework.CChart;

public class ChartFrame extends JFrame {

public static String DEFAULT\_TITLE = "Add Your Chart";

public static int DEFAULT\_WIDTH = 300;

public static int DEFAULT\_HEIGHT = 300;

private ChartPanel chartPanel;

private String chartTitle;

// Creates a window with a blank chart panel and default settings, such

// as size and title

public ChartFrame() {

super(DEFAULT\_TITLE);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.add(new ChartPanel());

this.setSize(DEFAULT\_WIDTH, DEFAULT\_HEIGHT);

this.setVisible(true);

}

// Allows for a custom title and prepared chart

public ChartFrame(String chartTitle, CChart aChart) {

super(chartTitle);

this.chartPanel = new ChartPanel(aChart);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.add(new ChartPanel(aChart));

this.setSize(DEFAULT\_WIDTH, DEFAULT\_HEIGHT);

this.setVisible(true);

}

// May further be overloaded for size as needed...

// May be extended to add control for, like data source and chart type

}

## ChartPanel.java

//ChartPanel.java

package gui;

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JPanel;

import framework.CChart;

public class ChartPanel extends JPanel {

private CChart theChart;

// You may add another attribute to store a background color, and

// overload the constructor to set a color for the panel

public ChartPanel(CChart aChart) {

setBackground(Color.WHITE);

this.theChart = aChart;

}

// Provided for easy generation of a blank panel with no chart

public ChartPanel() {

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

theChart.draw(g);

}

}

## AShape.java

//AShape.java

package framework;

import java.awt.Graphics;

// A is for "abstract"

// An abstract class that will be extended by specific shape classes. like

// the CLin class.

public abstract class AShape implements IDrawable {

// Defines a reference point (refX, refY), which can be used in

// subclasses, such as CLine, or CBox, to offset shapes to generate

// chart more easily.

protected int refX;

protected int refY;

AShape() {

}

abstract public void draw(Graphics g);

}

## CBar.java

//CBar.java

package framework;

import java.awt.\*;

// C is for "concrete", and also for "color" since your can set a line color

public class CBar extends AShape {

private int x1;

private Color lineColor = Color.BLACK;

public CBar(int x1, int offset, Color lineColor) {

this.x1 = x1;

this.lineColor = lineColor;

this.refX = offset;

}

@Override

public void draw(Graphics g) {

//rectangles

Graphics2D g2 = (Graphics2D) g;

g2.setColor(lineColor);

g2.fillRect(refX, 265-x1, 28, x1);

//text

Font font = new Font("Serif", Font.BOLD, 17);

g2.setFont(font);

g2.setColor(Color.BLACK);

String output = Integer.toString(x1);

g2.drawString(output, refX, 265-x1);

}

// Provides an example for overriding the toString() method defined in

// the Object class

public String toString() {

return "Line in color=" + lineColor;

}

}

## CChart.java

//CChart.java

package framework;

import java.awt.Graphics;

// C is for "concrete"

// A concrete class that is drawable, by implementing the IDrawable interface

// An instance of CChart encapsulate a number of AShape objects, each of

// which is Drawable.

public class CChart implements IDrawable {

private AShape[] chartComponents;

public String title;

public CChart(AShape[] shapes) {

this.chartComponents = shapes;

this.title = "Random Chart";

}

// When the Chart draws itself, it will iterate through all of its

// shapes and draw them in the order as stored in the array.

// You need to create the CChart object with the order in mind, since

// a shape painted later may cover up those painted before it.

@Override

public void draw(Graphics g) {

for (AShape aShape : chartComponents)

aShape.draw(g);

}

}

## Cline.java

//CLine.java

package framework;

import java.awt.\*;

// C is for "concrete", and also for "color" since your can set a line color

public class CLine extends AShape {

private int x1;

private int y1;

private int x2;

private int y2;

public int Xoffset;

private Color lineColor = Color.BLACK;

public CLine(int x1, int y1, int x2, int y2, Color lineColor) {

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

this.lineColor = lineColor;

}

@Override

public void draw(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

g2.setColor(lineColor);

g2.setStroke(new BasicStroke(3));

g2.drawLine(x1, 265-y1, x2, 265-y2);

}

// Provides an example for overriding the toString() method defined in

// the Object class

public String toString() {

return "Line in color=" + lineColor;

}

}

## CScatter.java

//CScatter.java

package framework;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

// C is for "concrete", and also for "color" since your can set a line color

public class CScatter extends AShape {

private int x1;

private int y1;

public int Xoffset;

private Color lineColor = Color.BLACK;

public CScatter(int x1, int y1, Color lineColor) {

this.x1 = x1;

this.y1 = y1;

this.lineColor = lineColor;

}

@Override

public void draw(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

g2.setColor(lineColor);

g2.fillOval(x1, 265-y1, 10, 10);

}

// Provides an example for overriding the toString() method defined in

// the Object class

public String toString() {

return "Line in color=" + lineColor;

}

}

## IDrawable.java

//IDrawable.java

package framework;

import java.awt.Graphics;

// An interface for anything that can be drawn to the graphical context

// Don't create the g yourself: it's available in the paintComponent() method

public interface IDrawable {

void draw(Graphics g);

}