情報科学演習及び実験1（Java演習）

IS2 6316047 出納 光

平成29年10月17日

1　折れ線グラフ

※　データの要素数が多いと横軸に入りきらない恐れがあるため、
10 個くらいと一般的な要素数を扱うものとする。

main 関数ではウィンドウの描画を行う。
ここでは主に背景色の指定のみを行なっている。

paintComponent 関数では、
まず始めに try 内でデータの読み込み行う。
data.txt を引数として中身を読み込み、
readLine で行の読み込み、split で各行でスペースごとに要素を区切る。
後々使用するため、データの要素の総数 x を data.length より求め、
parseInt で String 型で読み込まれたデータを int 型に変換する。

続いて、折れ線グラフを実装するために、
drawLine と fillRect を用い、横軸及び縦軸、グラフの描画範囲を指定する。
後々使用するため、for ループから max と min にデータの最大値、最小値を格納する。
次の for ループでは要素の数だけ再帰し、横軸のメモリを作る。
続く for ループでは再び要素の数だけ再帰し、横軸に要素の順番を表示する。

その次には for ループを 5 回だけ再帰させ、縦軸を 4 等分にするメモリを作る。その際に最大値の 1/4 ずつで指標を示す。
最後の for ループではデータの 2 つの要素の座標を drawLine で結ぶ。
このとき、「ある要素の値/全データ中の最大値」に縦軸の幅を掛けることでグラフを拡大縮小し、適切なサイズに調整している。

最後に catch より、ファイルが見つからなかったときのエラーである"NOT_FOUND"、
何らかの事情によりファイルが開けなかったエラー"CAN_NOT_OPEN"を返す。

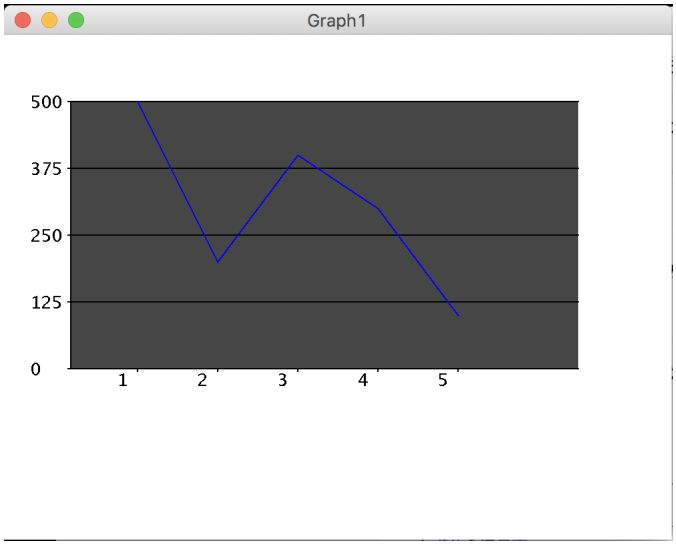

図 1: 実行例 1

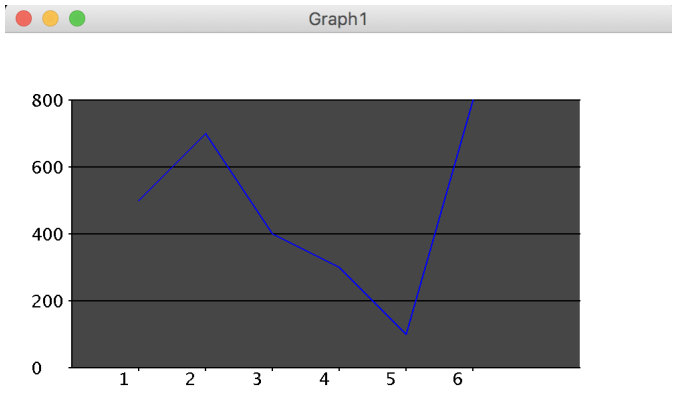

図 2: 実行例 2

グラフとして実用的に用いられる品質で描画されている。
それぞれの指標に対して、データの各値は適切な割合を示している。
データの要素数を増やしても、同様に維持される。
データのそれぞれの値の帯域を拡げても、同様に維持される。

2　円グラフ

main 関数ではウィンドウの描画を行う。
ここでは主に背景色の指定のみを行なっている。

paintComponent 関数では、
まず始めに try 内でデータの読み込み行う。
data.txt を引数として中身を読み込み、
readLine で行の読み込み、split で各行でスペースごとに要素を区切る。
後々使用するため、データの要素の総数 x を data.length より求め、
parseInt で String 型で読み込まれたデータを int 型に変換する。

続いて、円グラフを実装するために、
まず後々用いることとなるため、for ループにより全データの合計値 sum を求める。
また開始角 start を 90 度に指定し、円グラフが最上部から開始するように設定する。
その後、for ループによりデータの要素の数だけ再帰させる。その中で、まずは「ある要素の値/全データの合計値」より、その要素が占める割合 rate を示し、
それに 360 を掛けることでデータの要素 1 つあたりが占める角度 rad を求める。
続いて、setColor の引数に 0 から 255 までの数値をランダムに RGB として割り当てる。
fillArc の引数として先ほど用意した start を開始角として、rad を描画角として割り当て、円グラフを描画する。
このとき、円グラフは時計回りであるため、-rad として引数を取ることに注意した。
円グラフの描画の最後に開始角に描画角を足して、次の開始角が続きから始まるようにする。

続く drawString では、データの各値を sin 関数と cos 関数を用いることで円の周囲に表示する。
さらにどの色がどの要素であるかを示す指標も添付した。
このときには、ランダムで変化する色を setColor により黒色で表示することに気をつけた。

最後に catch より、ファイルが見つからなかったときのエラーである"NOT_FOUND"、何らかの事情によりファイルが開けなかったエラー"CAN_NOT_OPEN"を返す。

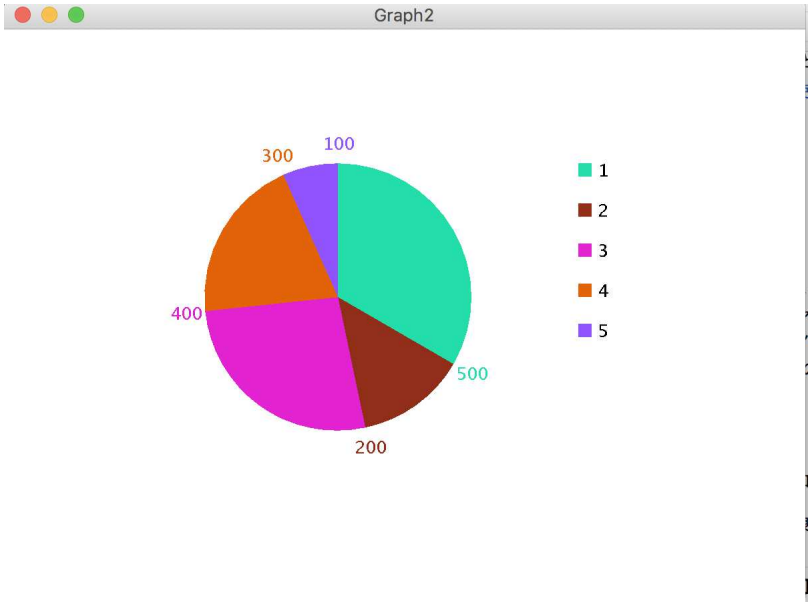

図 3: 実行例 1

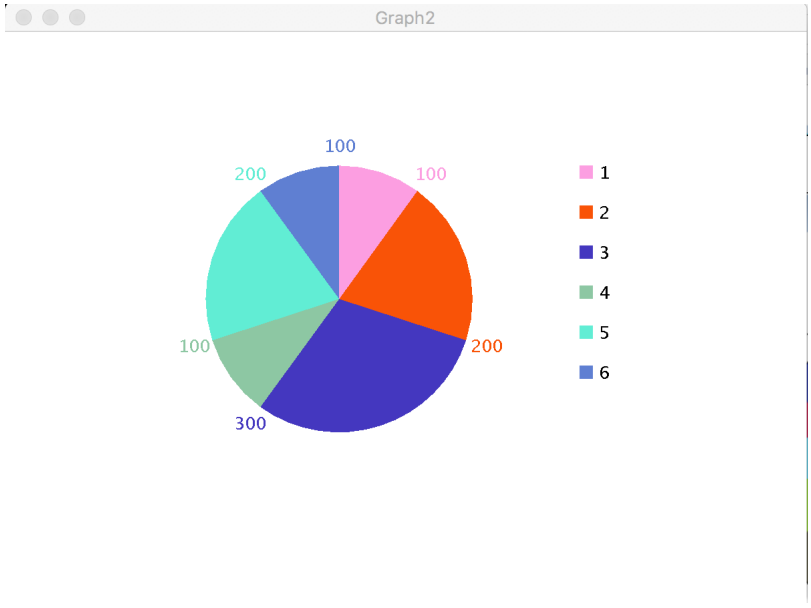

図 4: 実行例 2

グラフとして実用的に用いられる品質で描画されている。
それぞれの指標に対して、データの各値は適切な割合を示している。

データの要素数を増やしても、同様に維持される。
データのそれぞれの値の帯域を拡げても、同様に維持される。

3　レーダーチャート

main 関数ではウィンドウの描画を行う。
ここでは主に背景色の指定のみを行なっている。

paintComponent 関数では、
まず始めに try 内でデータの読み込み行う。
data.txt を引数として中身を読み込み、
readLine で行の読み込み、split で各行でスペースごとに要素を区切る。
catch より、ファイルが見つからなかったときのエラーである"NOT_FOUND"、
何らかの事情によりファイルが開けなかったエラー"CAN_NOT_OPEN"を返す。

続いて、レーダーチャートを実装するために、
後々用いるため、まずデータの要素の総数 x を data.length より求め、
parseInt で String 型で読み込まれたデータを int 型に変換する。
for ループから max と min にデータの最大値、最小値を格納する。
続いてデータの各要素の位置を表示するために、mtx にその x 座標、mty にその y 座標を配列として割り当てる。
その引数には、「データの各要素の値/全データの最大値」にグラフの描画範囲を幅を掛けることでグラフを拡大縮小し、適切なサイズに調整している。
setColor により色を変更し、fillPolygon を用いて各要素の占める範囲を描画する。
ここでは、先に範囲を描画し、その後グラフのメモリを描画するという順番で実装しなければ、
実行結果で表示されるグラフの範囲にメモリが上書きされ、見辛くなってしまう点に注意した。

最後に、5 段階のメモリに最大値の 1/5 が指標として表示されるように実装した。

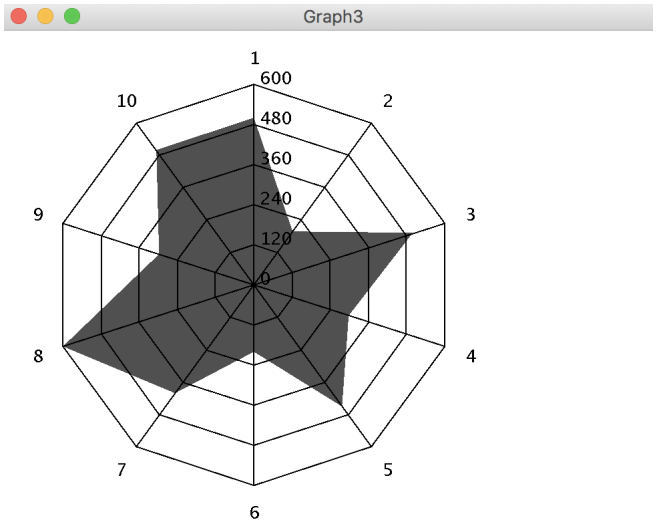

図 5: 実行例 1

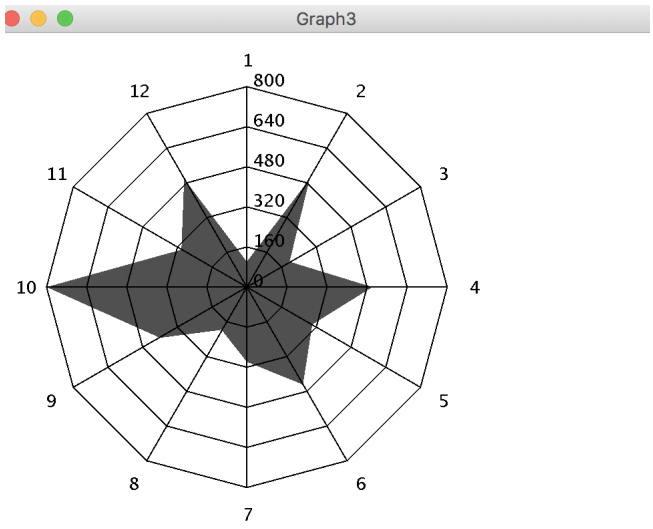

図 6: 実行例 2

グラフとして実用的に用いられる品質で描画されている。
それぞれの指標に対して、データの各値は適切な割合を示している。
データの要素数を増やしても、同様に維持される。
データのそれぞれの値の帯域を拡げても、同様に維持される。

4　考察

今後のチームで開発していく際の長期的なメンテナンスも見据え、ソースコードに十分な可読性を持たせるためにも、
「分かり易いクラス名及び変数名をつけること」
「コメントアウトを用い補足説明を加えること」
「エラーを状態に応じて細かく分けること」
といったことは勿論、
「コンポーネント単位でまとまりを作ること」や「明示的に変数を宣言すること」が大切である。