

第14回

まとめ

前回宿題

- 以下のプログラムを作成せよ
課題13-1: 円グラフの作成 (makePieChart)
課題13-2: 棒グラフの作成 (makeBarChart)

課題13-1(円グラフの作成)

2021年の東京の月別降水量(アメダス月別値の一部、単位はmm)が1月から12月まで順に `public static double[] data` の配列に格納されている。この月別降水量から円グラフを作成する以下のメソッドを完成させよ

```
void makePieChart()
```

円グラフ作成条件は以下の通り

- 円グラフには、インスタンス化されている `pieChart` を使用する
- 円グラフの真上を起点にして、1月、2月、... 12月の順に時計回りに降水量を配置する
- 各区分の名前は、「1月」というような文字列にする(数字は半角)
- 各区分の値は、配列に格納されている実数値をそのまま使用する
- 凡例をグラフの右側に表示する
- その他、区分の色、文字の大きさ等のデザインは任意

解答例

```
void makePieChart() {  
    // 課題 13-1 のコード  
    PieChart.Data[] pieChartData = new PieChart.Data[12];  
    for (int i = 0; i < 12; i++) {  
        pieChartData[i] = new PieChart.Data((i + 1) + "月", data[i]);  
    }  
  
    pieChart.getData().addAll(pieChartData);  
    pieChart.setTitle("2021年東京");  
    pieChart.setClockwise(true);  
    pieChart.setStartAngle(90);  
    pieChart.setLegendSide(Side.RIGHT);  
}
```

課題13-2(棒グラフの作成)

課題13-1と同じデータを用いて棒グラフを作成する以下のメソッドを作成せよ

```
void makeBarChart()
```

棒グラフ作成条件は以下の通り

- 棒グラフには、インスタンス化されている `barChart`, `xAxis`, `yAxis` を使用する
- 縦棒グラフとし、左側から1月、2月、... 12月の順に降水量を配置する
- 横軸は、「1月」というような文字列にする(数字は半角)
- 縦軸は、配列に格納されている実数値をそのまま使用する
- 縦軸には、「降水量(mm)」という形式のラベルを付ける
- 棒の色、棒の間隔、文字の大きさ等のデザインは任意

作成したメソッドの動作は、テストプログラム `RailfallGraphTester` を使って確認すること

解答例

```
void makeBarChart() {  
    // 課題 13-2 のコード  
    XYChart.Series<String, Number> series = new XYChart.Series<>();  
    series.setName("2021年東京");  
    for (int i = 0; i < 12; i++) {  
        series.getData().add(new XYChart.Data((i + 1) + "月", data[i]));  
    }  
  
    barChart.getData().add(series);  
    barChart.setLegendSide(Side.TOP);  
    xAxis.setTickLength(0);  
    yAxis.setLabel("降水量 (mm)");  
}
```

TableViewクラス (javafx.scene.control TableView)

<https://docs.oracle.com/javase/jp/8/javafx/api/javafx/scene/control/TableView.html>

The screenshot shows the Oracle JavaFX 8 API documentation for the `TableView` class. The browser window title is "TableView (JavaFX 8)". The URL bar shows the link to the documentation page. The left sidebar contains a navigation menu with "All Classes", "パッケージ" (Packages), and "すべてのクラス" (All Classes). The "すべてのクラス" section is expanded, showing a list of classes including `AccessibleAction`, `AccessibleAttribute`, `AccessibleRole`, `Accordion`, `ActionEvent`, `Affine`, `Alert`, `Alert.AlertType`, `AmbientLight`, `AnchorPane`, `Animation`, `Animation.Status`, `AnimationTimer`, `Application`, `Application.Parameters`, `Arc`, `ArcTo`, `ArcType`, and `AreaChart`. The main content area displays the class hierarchy for `TableView`, starting from `java.lang.Object` and going down to `javafx.scene.control.Parent`, `javafx.scene.layout.Region`, `javafx.scene.control.Control`, and finally `javafx.scene.control TableView<S>`. Below the hierarchy, the "型パラメータ:" (Type Parameters) section explains that `S` is the type of objects contained in the list. The "すべての実装されたインタフェース:" (All Implemented Interfaces) section lists `Styleable`, `EventTarget`, and `Skinnable`. The "ソースコード" (Source Code) section shows the class declaration: `@DefaultProperty(value="items") public class TableView<S> extends Control`. A descriptive paragraph follows, stating that `TableView` is designed for visualizing unlimited data rows and columns, and that it is similar to `ListView` but also supports columns. It references the "TableViewの作成" (Creating TableView) section for more details.

TableColumnクラス (javafx.scene.control.TableColumn)

<https://docs.oracle.com/javase/jp/8/javafx/api/javafx/scene/control/TableColumn.html>

The screenshot shows the Oracle JavaFX 8 API documentation for the `TableColumn` class. The left sidebar lists various JavaFX classes and packages. The main content area displays the class hierarchy, type parameters, and a brief description of the class.

JavaFX 8

All Classes

パッケージ

- javafx.animation
- javafx.application
- javafx.beans

すべてのクラス

- AccessibleAction
- AccessibleAttribute
- AccessibleRole
- Accordion
- ActionEvent
- Affine
- Alert
- Alert.AlertType
- AmbientLight
- AnchorPane
- Animation
- Animation.Status
- AnimationTimer
- Application
- Application.Parameters
- Arc
- ArcTo
- ArcType
- AreaChart

概要 パッケージ クラス 使用 階層ツリー 非推奨 索引 ヘルプ

前のクラス 次のクラス フレーム フレームなし

サマリー: ネスト | フィールド | コンストラクタ | メソッド 詳細: フィールド | コンストラクタ | メソッド

javafx.scene.control

クラス `TableColumn<S,T>`

java.lang.Object
javafx.scene.control.TableColumnBase<S,T>
javafx.scene.control.TableColumn<S,T>

型パラメータ:

- S - `TableView`汎用型の型(つまり、`S == TableView<S>`)
- T - この`TableColumn`のすべてのセルのコンテンツの型。

すべての実装されたインタフェース:

Styleable、EventTarget

```
public class TableColumn<S,T>
    extends TableColumnBase<S,T>
    implements EventTarget
```

`TableView`は、多数の`TableColumn`インスタンスで構成されます。表内の各`TableColumn`では、その列のコンテンツを表示(および編集)します。`TableColumn`では1つの列のデータの表示および編集を行うことに加え、次のことに必要なプロパティも含まれています。

- サイズ変更(`minWidth/prefWidth/maxWidth`および`width`プロパティを使用)
- visibilityの切替え
- header textの表示

TableViewクラスの使い方(1/3)

➤ TableColumnの生成

- カラム名 (String) を指定してデータカラムを生成する

```
TableColumn firstNameCol = new TableColumn("First Name");
```

➤ TableViewの生成

- 空のテーブルを生成する

```
TableView table = new TableView();
```

- テーブルにテーブルカラムを複数件追加する

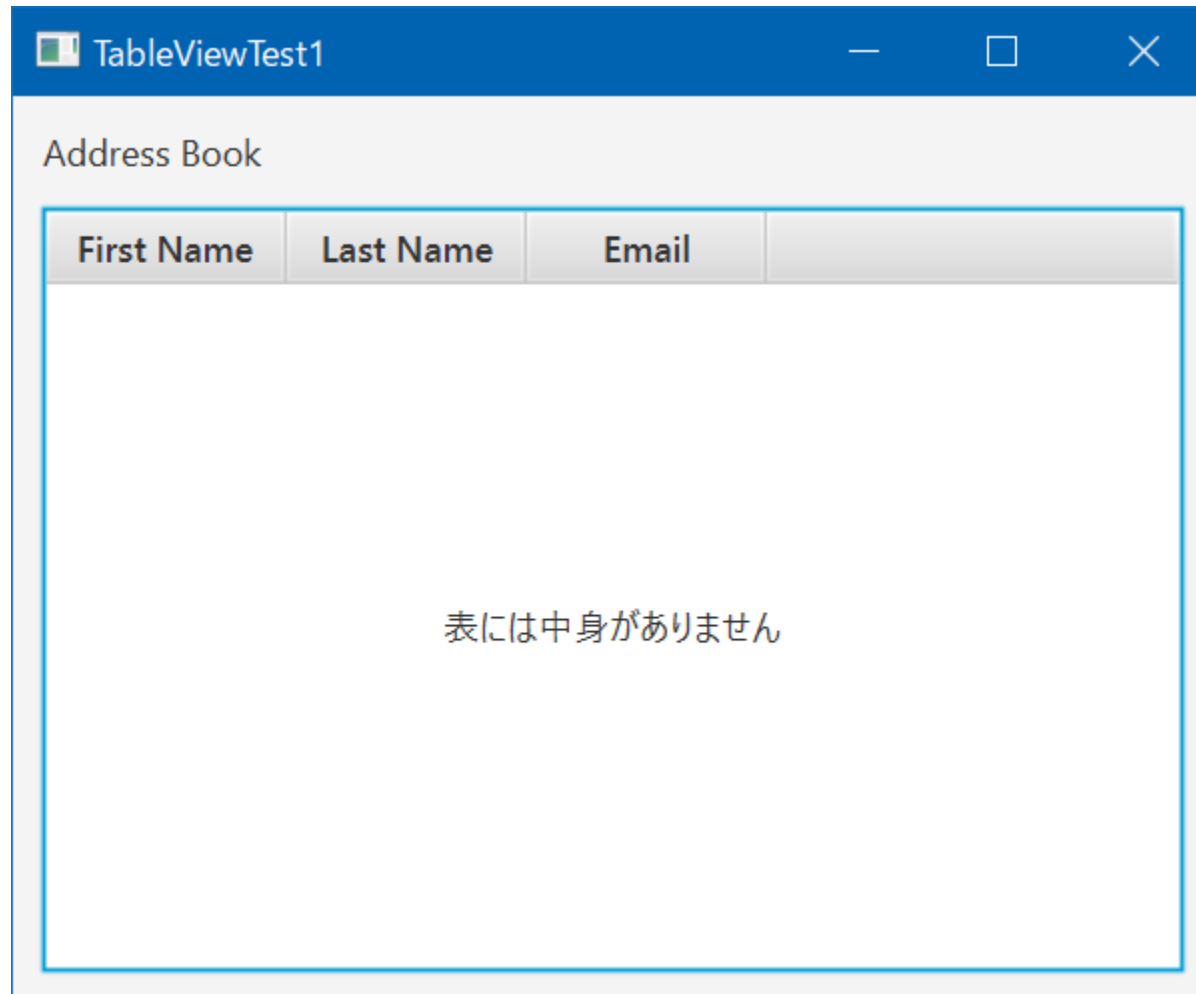
```
table.getColumns().addAll(firstNameCol, ...);
```

備考: 実際に使用する際には、型を指定する必要がある。なお、これだけでは、テーブルにデータを表示することはできない。

Listing 14-1: TableViewTest1.java

```
public void start(Stage stage) {  
  
    Label label = new Label("Address Book");  
  
    TableColumn firstNameCol = new TableColumn("First Name");  
    TableColumn lastNameCol = new TableColumn("Last Name");  
    TableColumn emailCol = new TableColumn("Email");  
    table.getColumns().addAll(firstNameCol, lastNameCol, emailCol);  
  
    VBox vbox = new VBox(10, label, table);  
    vbox.setPadding(new Insets(10));  
    Scene scene = new Scene(vbox, 400, 300);  
    stage.setScene(scene);  
    stage.setTitle("TableViewTest1");  
    stage.show();  
}
```

TableViewTest1



FXCollectionsクラス (javafx.collections.FXCollections)

<https://docs.oracle.com/javase/jp/8/javafx/api/javafx/collections/FXCollections.html>

The screenshot shows the Oracle JavaFX 8 API documentation for the `FXCollections` class. The browser window title is "FXCollections (JavaFX 8)". The URL bar shows the link to the documentation page. The left sidebar lists the package hierarchy: `javafx.animation`, `javafx.application`, and `javafx.beans`. The main content area displays the class `FXCollections` as a utility class that extends `Object`. It includes a summary of its static methods, which are 1:1 copies of the `java.util.Collections` methods. The documentation also mentions that these methods are optimized for performance by generating notifications only when necessary. The version of JavaFX is listed as 2.0. The bottom section of the page is titled "メソッドのサマリー" (Method Summary).

JavaFX 8

All Classes

パッケージ

javafx.animation
javafx.application
javafx.beans

すべてのクラス

AccessibleAction
AccessibleAttribute
AccessibleRole
Accordion
ActionEvent
Affine
Alert
Alert.AlertType
AmbientLight
AnchorPane
Animation
Animation.Status
AnimationTimer
Application
Application.Parameters
Arc
ArcTo
ArcType
AreaChart

概要 パッケージ クラス 使用 階層ツリー 非推奨 索引 ヘルプ

前のクラス 次のクラス フレーム フレームなし

サマリー: ネスト | フィールド | コンストラクタ | メソッド 詳細: フィールド | コンストラクタ | メソッド

javafx.collections

クラスFXCollections

java.lang.Object
javafx.collections.FXCollections

public class **FXCollections**
extends Object

java.util.Collectionsメソッドの1対1のコピーであるstaticメソッドで構成されるユーティリティ・クラス。

ラッパー・メソッド(synchronizedObservableListまたはemptyObservableListなど)の機能は、Collectionsのメソッドとまったく同じです。ただし、ObservableListを返すため、入力でObservableListを必要とするメソッドに適しているという点が異なります。

ユーティリティ・メソッドはここでは主にパフォーマンス上の理由で使用されます。すべてのメソッドは限られた数の通知のみを生成するという方法で最適化されます。一方、java.util.Collectionsメソッドは、ObservableListで変更メソッドを複数回呼び出すことができるため、複数の通知が生成されます。

導入されたバージョン:
JavaFX 2.0

メソッドのサマリー

PropertyValueFactoryクラス (javafx.scene.control.cell.PropertyValueFactory)

<https://docs.oracle.com/javase/jp/8/javafx/api/javafx/scene/control/cell/PropertyValueFactory.html>

The screenshot shows the Oracle JavaFX 8 API documentation for the `PropertyValueFactory` class. The left sidebar lists the package `javafx.scene.control.cell` and the class `PropertyValueFactory`. The main content area shows the class hierarchy, type parameters, and a code example.

Class Hierarchy:

- `java.lang.Object`
- `javafx.scene.control.cell.PropertyValueFactory<S,T>`

型パラメータ:

- `S` - `TableView.items` リストに含まれるクラスの型。
- `T` - `TableColumn` セルに含まれるクラスの型。

すべての実装されたインタフェース:

- `Callback<TableColumn.CellDataFeatures<S,T>,ObservableValue<T>>`

Code Example:

```
public class PropertyValueFactory<S,T>
    extends Object
    implements Callback<TableColumn.CellDataFeatures<S,T>,ObservableValue<T>>

    TableColumn<Person,String> firstNameCol = new TableColumn<Person,String>("First Name");
    firstNameCol.setCellValueFactory(new PropertyValueFactory<Person,String>("firstName"));
```

この例では、`firstName` 文字列が、`Person` クラスの型 (`TableView.items` リストのクラスの型) の想定された

TableViewクラスの使い方(2/3)

➤ テーブルとテーブルカラムの型を指定

- TableView<Person> table = new TableView<>();
- TableColumn<Person, String> firstNameCol =
new TableColumn<>("First Name");
firstNameCol.setCellValueFactory(
new PropertyValueFactory<>("firstName")); ※ゲッターが必要

➤ テーブルに表示するデータの生成

- FXCollections.observableArrayListに格納する
ObservableList<Person> data =
FXCollections.observableArrayList();

➤ テーブルにデータを格納

- table.setItems(data);

Listing 14-2: Person.java

```
public class Person {  
  
    private final String firstName;  
    private final String lastName;  
    private final String email;  
  
    public Person(String fName, String lName, String email) {  
        this.firstName = fName;  
        this.lastName = lName;  
        this.email = email;  
    }  
  
    public String getFirstName() {  
        return firstName;  
    }  
  
    public String getLastName() {  
        return lastName;  
    }  
  
    public String getEmail() {  
        return email;  
    }  
}
```

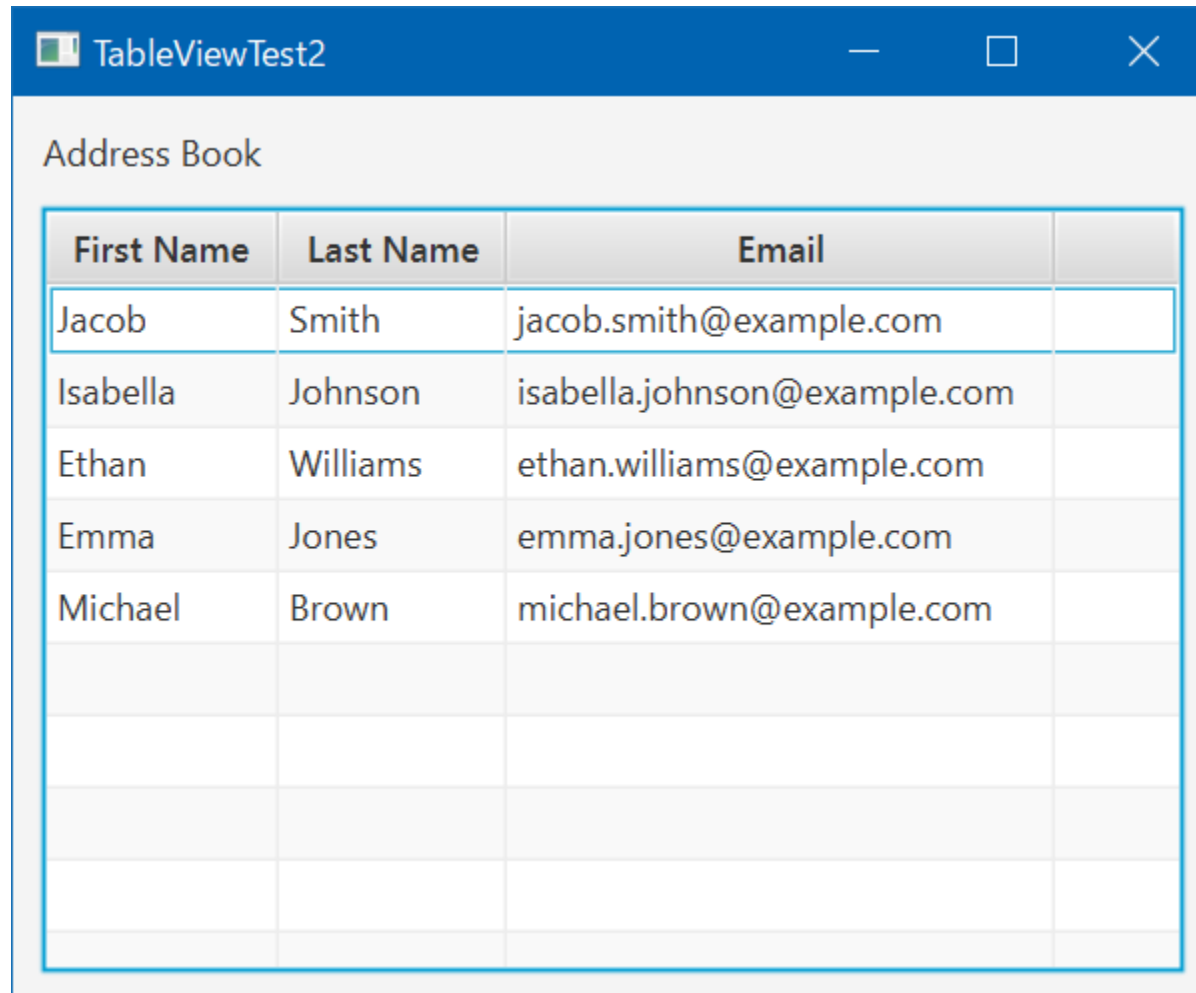
Listing 14-3: TableViewTest2.java

```
TableView<Person> table = new TableView<>();
ObservableList<Person> data = FXCollections.observableArrayList(
    new Person("Jacob", "Smith", "jacob.smith@example.com"),
    new Person("Isabella", "Johnson", "isabella.johnson@example.com"),
    new Person("Ethan", "Williams", "ethan.williams@example.com"),
    new Person("Emma", "Jones", "emma.jones@example.com"),
    new Person("Michael", "Brown", "michael.brown@example.com"));

Label label = new Label("Address Book");

TableColumn<Person, String> firstNameCol = new TableColumn<>("First Name");
firstNameCol.setCellValueFactory(new PropertyValueFactory<>("firstName"));
TableColumn<Person, String> lastNameCol = new TableColumn<>("Last Name");
lastNameCol.setCellValueFactory(new PropertyValueFactory<>("lastName"));
TableColumn<Person, String> emailCol = new TableColumn<>("Email");
emailCol.setCellValueFactory(new PropertyValueFactory<>("email"));
table.getColumns().addAll(firstNameCol, lastNameCol, emailCol);
table.setItems(data);
```


TableViewTest2



TableViewTest2

Address Book

First Name	Last Name	Email	
Jacob	Smith	jacob.smith@example.com	
Isabella	Johnson	isabella.johnson@example.com	
Ethan	Williams	ethan.williams@example.com	
Emma	Jones	emma.jones@example.com	
Michael	Brown	michael.brown@example.com	

TableViewクラスの使い方(3/3)

➤ データ(行)の追加

- データ表示用の ObservableList に要素を追加すればよい
- 例えば、テキストフィールドに入力されたデータを追加する場合は、
`data.add(new Person(addFirstName.getText(),
addLastName.getText(), addEmail.getText()));`

➤ データ(行)の削除

- データ表示用の ObservableList から要素を削除すればよい
- どの要素を削除すれば良いかは、getSelectedIndexメソッドで取得
- 例えば、ユーザが選択している行を削除する場合は、
`int index = table.getSelectionModel().getSelectedIndex();
data.remove(index);`

Listing 14-4: TableViewTest3.java (1/2)

```
TableColumn<Person, String> firstNameCol = new TableColumn<>("First Name");  
firstNameCol.setCellValueFactory(new PropertyValueFactory<>("firstName"));  
TableColumn<Person, String> lastNameCol = new TableColumn<>("Last Name");  
lastNameCol.setCellValueFactory(new PropertyValueFactory<>("lastName"));  
TableColumn<Person, String> emailCol = new TableColumn<>("Email");  
emailCol.setCellValueFactory(new PropertyValueFactory<>("email"));  
table.getColumns().addAll(firstNameCol, lastNameCol, emailCol);  
table.setItems(data);
```

```
TextField addFirstName = new TextField();  
addFirstName.setMaxWidth(firstNameCol.getPrefWidth());  
addFirstName.setPromptText("First Name");  
TextField addLastName = new TextField();  
addLastName.setMaxWidth(lastNameCol.getPrefWidth());  
addLastName.setPromptText("Last Name");  
TextField addEmail = new TextField();  
addEmail.setMaxWidth(emailCol.getPrefWidth());  
addEmail.setPromptText("Email");
```

Listing 14-4: TableViewTest3.java (2/2)

```
Button addButton = new Button("Add");
addButton.setOnAction((ActionEvent e) -> {
    if (!addEmail.getText().isEmpty()) {
        data.add(new Person(
            addFirstName.getText(),
            addLastName.getText(),
            addEmail.getText()));
        addFirstName.clear();
        addLastName.clear();
        addEmail.clear();
    }
});

Button deleteButton = new Button("Delete");
deleteButton.setOnAction((ActionEvent e) -> {
    int index = table.getSelectionModel().getSelectedIndex();
    if (index >= 0) {
        data.remove(index);
    }
});
```

TableViewTest3

TableViewTest3

Address Book

First Name	Last Name	Email	
Jacob	Smith	jacob.smith@example.com	
Isabella	Johnson	isabella.johnson@example.com	
Ethan	Williams	ethan.williams@example.com	
Emma	Jones	emma.jones@example.com	
Michael	Brown	michael.brown@example.com	
Ushio	Inoue	inoueu@mail.dendai.ac.jp	

First Name Last Name Email Add Delete

オブジェクト指向プログラミングの本質は何か？

➤ 部品化

- データと手続きを一体化したオブジェクトで管理
- オブジェクトへのアクセス方法を標準化(カプセル化)

➤ 再利用

- 既存のクラスから新しいクラスを派生(資産の継承)
- 追加するデータと手続きのみを記述(差分プログラミング)

➤ 柔軟性

- 異なる種類の部品の一元的な管理(多相性)
- オブジェクトの振る舞いは実行時に自身が決定(動的束縛)

第3回レポート

プロローグ

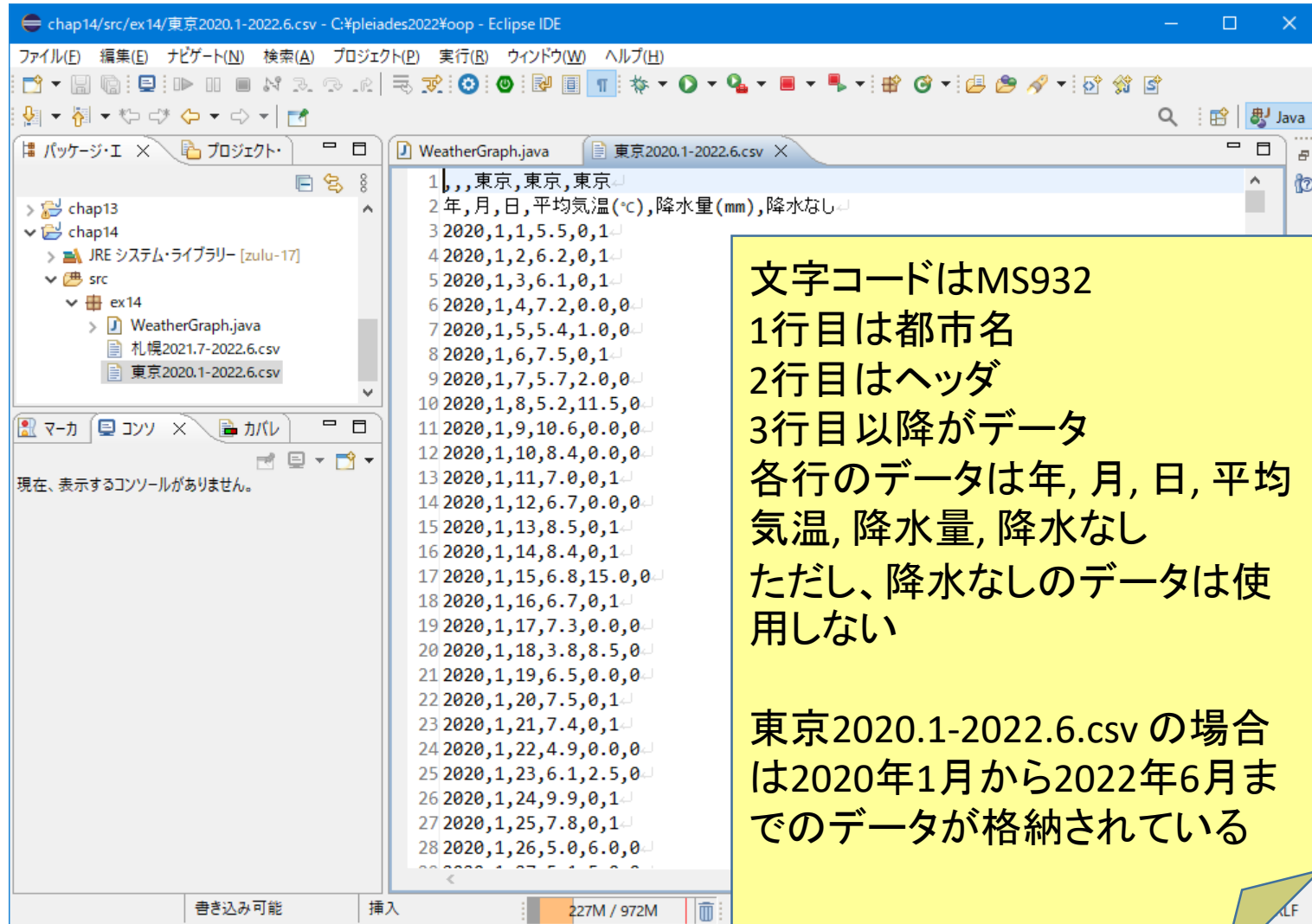
気象庁は日本全国の気象データを1時間ごとにホームページで自動更新している(<https://www.jma.go.jp/jp/amedas/>)。また、過去の気象データをダウンロードするサービスも行っている。演習13では、東京の昨年1年間の月別の降水量データを用いて、円グラフと棒グラフを作成した。

第3回レポートでは、CSV形式のファイルに格納されたある都市の過去約数年分の日別の平均気温と降水量データを用いて、ユーザが指定した月の折れ線グラフと棒グラフを作成する。

課題の内容

1. 以下のGUIアプリケーションをJavaFXで作成せよ。ソースファイル WeatherGraph.java を使用すること。
 - グラフを作成するためのデータは、CSV形式のファイルに格納されている。ファイルの先頭行に都市名、2行目からは1日1行の形式で、年, 月, 日, 平均気温, 降水量, 降水なしフラグの順にカンマ区切りで格納されている。なお、データは、年, 月, 日の昇順に欠落なく規則正しく格納されている。
 - ファイルの場所は、FileChooserクラスを用いてユーザに選択させる。
 - 画面に表示するのは、ユーザが指定した年・月の各日のデータを用いた2次元グラフであり、横軸を日付、縦軸を平均気温とする折れ線グラフと、横軸を日付、縦軸を降水量とする棒グラフである。
 - ユーザは指定した年・月を変更して、繰り返しグラフを表示できる。さらに、ファイルを選択し直すこともできる。なお、現在グラフで表示している都市名と年・月を明示すること。
 - 画面のレイアウト、使用するGUI部品は任意のものを使用してよい。また、CSSを使用する場合には、ソースファイルと同じフォルダに配置すること。

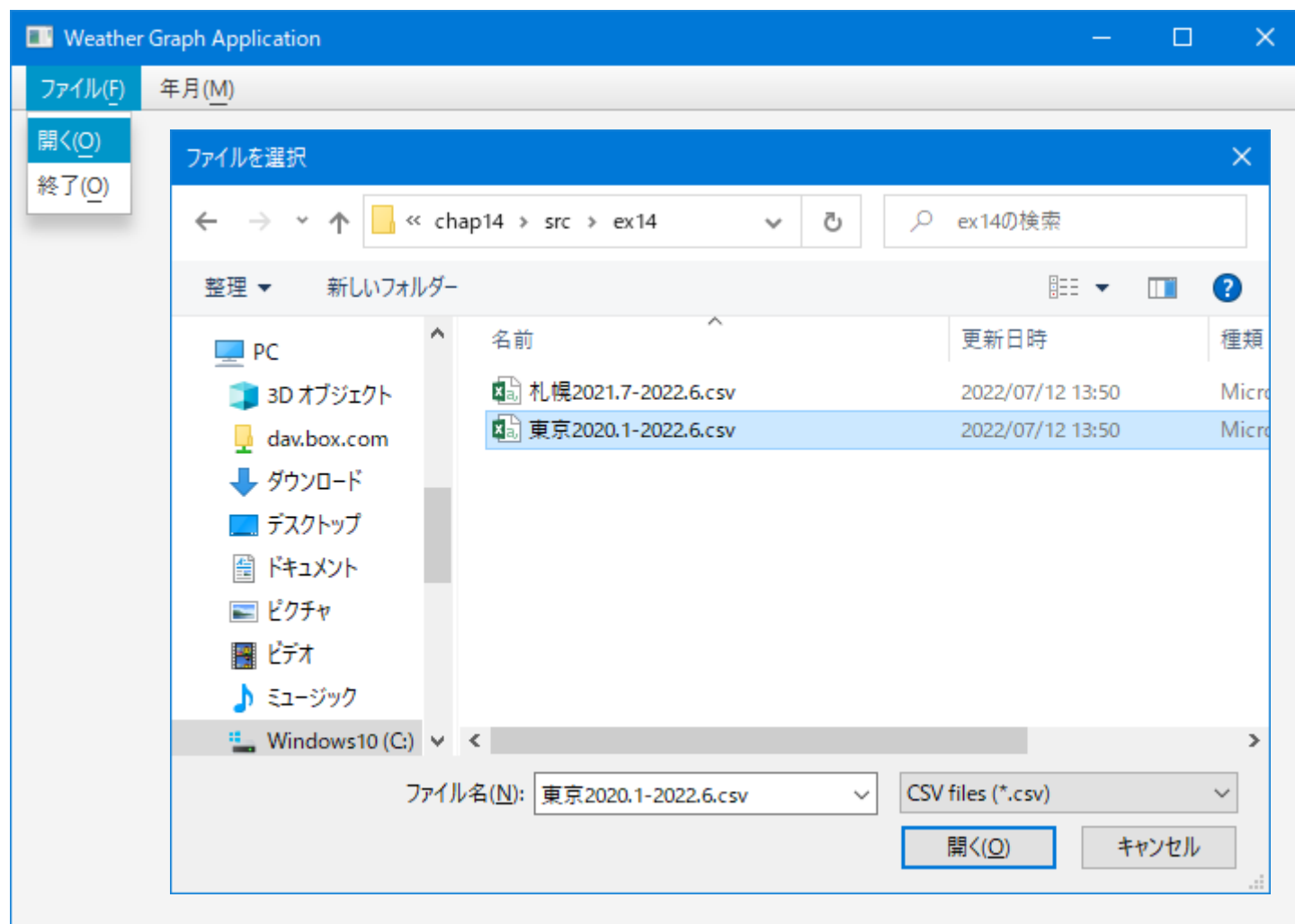
CSV形式ファイルの内容



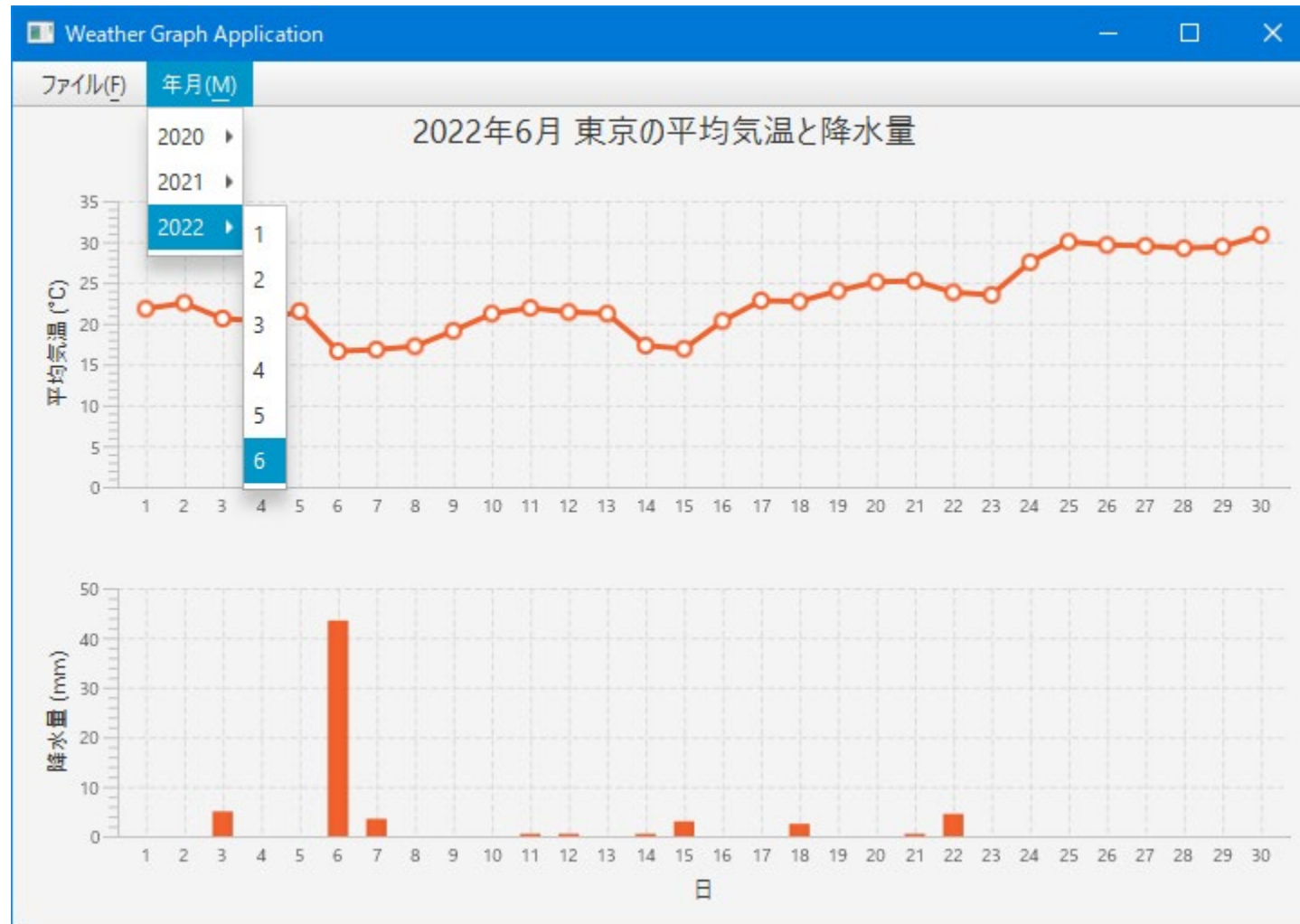
文字コードはMS932
1行目は都市名
2行目はヘッダ
3行目以降がデータ
各行のデータは年, 月, 日, 平均
気温, 降水量, 降水なし
ただし、降水なしのデータは使
用しない

東京2020.1-2022.6.csv の場合
は2020年1月から2022年6月ま
でのデータが格納されている

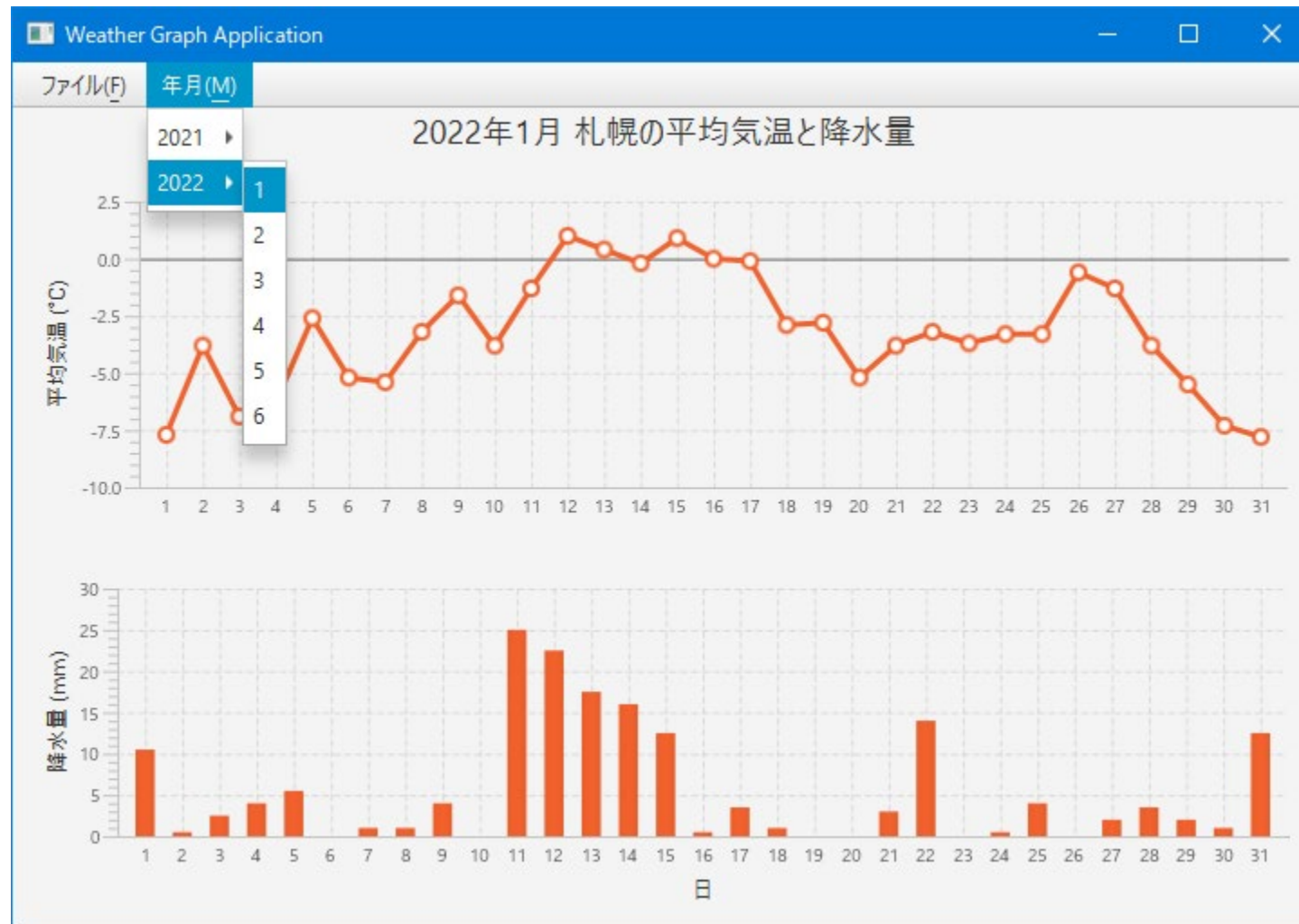
GUIの動作イメージ(使用するレイアウトと部品は任意)



GUIの動作イメージ(その2)



GUIの動作イメージ(その3)



課題の内容(続き)

2. 前記のGUIアプリケーション作成において、以下の考察をソースファイル WeatherGraph.java 中にコメントとして200～300文字程度の日本語文章で記述せよ。
- 自分で調査したこと
 - 特に工夫または苦心したこと
 - プログラムの動作を確認したこと

完成したソースファイル WeatherGraph.java は、WebClassの14レポート第3回のページにアップロードせよ。

提出期限は、7月22日(金) 23:59とする。

WeatherGraph.java の内容

```
chp14/src/ex14/WeatherGraph.java - C:\pleiades2022\oop - Eclipse IDE
ファイル(E) 編集(E) ソース(S) リファクタリング(R) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)

パッケージ・エクスプローラ
  chap13
  chap14
    JRE システム・ライブラリー [zulu-17]
    src
      ex14
        WeatherGraph.java
        札幌2021.7-2022.6.csv
        東京2020.1-2022.6.csv

コンソール
  現在、表示するコンソールがありません。

WeatherGraph.java
4 import javafx.application.Application;
6
7 public class WeatherGraph extends Application {
8
9     public static String gakuban = "20EC000"; // 学籍番号を入力すること
10    public static String yourname = "千住旭"; // 氏名を入力すること
11
12    @Override
13    public void start(Stage primaryStage) {
14        // プログラムを作成
15
16        primaryStage.show();
17    }
18
19    public static void main(String[] args) {
20        // アプリケーションを起動する
21        Application.launch(args);
22        System.out.println("完了 " + gakuban + " " + yourname);
23    }
24
25 }
26
27 /* 考察 -- 調査したこと、工夫したこと、確認したことを記述
28
29
30
31
32 */
```

学籍番号と氏名を入力

GUIアプリケーションのソースコード

アプリケーション作成におけるコメント

注意事項

1. UTF-8形式のソースファイルを提出すること。
2. ソースファイルのコンパイル、実行が正しくできることを十分に確認してから提出すること。
3. ソースファイル中のコメント欄に、考察を必ず記述すること。
考察がない、または不十分なものは、プログラムが正しく出来ていても大幅な減点となる。
4. 他人のソースファイルをコピーしたり、コピーさせたりしないこと。
もし、コピーしたと思われる酷似したファイルが提出された場合は、評価の対象外とする。自分で意図せずに他人にコピーされた場合も同様なので、ファイルの取り扱いについて十分に注意すること。