COMP10062: Assignment 5 v103

In this assignment you will build an app to redeem tickets at a comic book show. The java ***Ticket Scanning App*** will scan tickets for people entering the show. See Figure 1, for the app's GUI interface. This is a real-world app that can be used to redeem tickets at any type of consumer show. Figure 2., shows you what a ticket looks like. The ticket's barcode and its equivalent text are shown underneath the barcode.

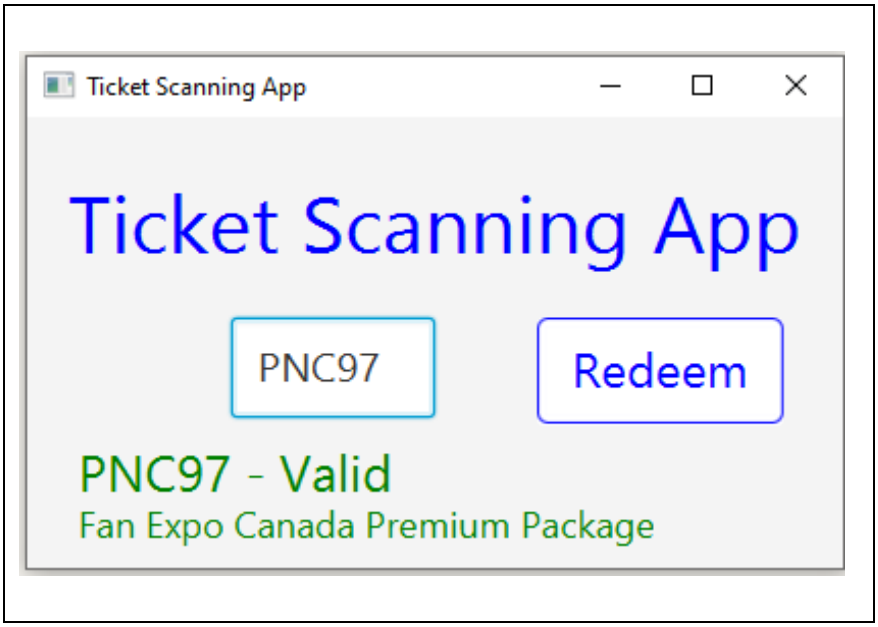


Figure 1. Product shot of the Ticketing Scanning App

Figure 2. A Sample Ticket

The universe of all tickets belonging to the comic book show are found in a TAB delimited file called, ***codes.txt***. Figure 3., shows how the four columns in ***codes.txt*** are arranged.

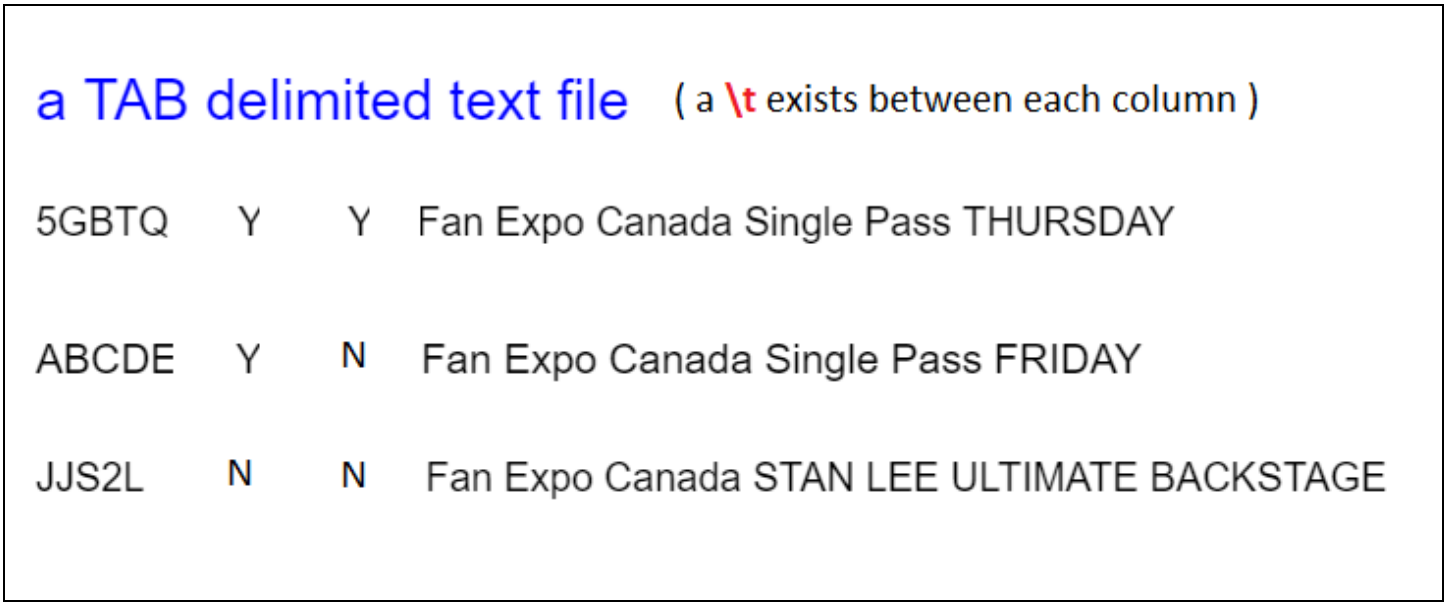


Figure 3. The ***codes.txt*** file.

The ***codes.txt*** file contains a line for each valid show ticket. Each ticket admits only one person. The state of each ticket varies.

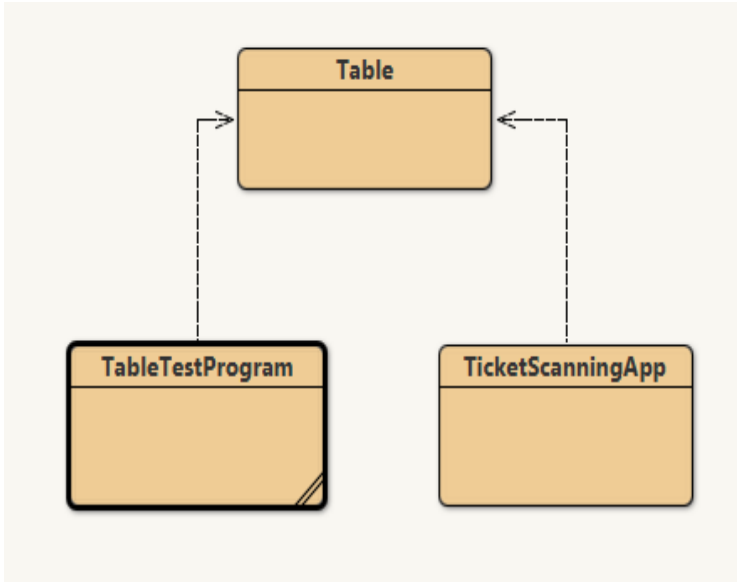
Column 0: is the unique barcode of the ticket, this is a primary key.
Column 1: is Y or N depending on whether the ticket has been purchased
Column 2: is Y or N depending on whether or not the ticket holder has entered the show
Column 3: is the event's name

An example ***codes.txt*** file

ABCDE	N	N	Fan Expo Canada Single Pass FRIDAY
XQ34Y	Y	N	Fan Expo Canada Single Pass FRIDAY
EZCMN	Y	Y	Fan Expo Canada Single Pass FRIDAY

ABCDE is a ticket which has not been purchased yet.
XQ34Y is a ticket which has been purchased and the person has not entered the show yet.
EZCMN is a ticket which has been purchased and the person has entered the show.
123AA is an invalid ticket, because it does not appear in the example, ***codes.txt*** file

Note: if EZCMN is scanned again for a second time, the App should respond with '**Duplicate**' as shown in Figure 7, to indicate that this code has already been scanned. The person holding this ticket would not be admitted into the show.



Important Points

1. Each unique barcoded ticket admits only **one** person into the show.

2. The ***codex.txt*** file contains the full universe of all possible show tickets. That is, tickets that are not bought yet, tickets that are bought but unredeemed, and redeemed tickets. Any code not in the ***codes.txt*** file is invalid.

Figure 4. Java Class Layout

About the Table Class

The Table class solely manages the ***codes.txt*** tab delimited file. Its UML is shown in Figure 11. This class can manage any tab delimited file and has no dependency on our ticket application. The starter kit provides an implementation of the Table class's constructor. The rest of the Table class methods need to be built.

About the TableTestProgram class

The TableTestProgram class is a test program which tests just the Table class. The TableTestProgram should display a menu like or similar to the menu shown in Figure 12. The TableTestProgram has no dependencies or role in the ticket scanning app.

About the TicketScanningApp class

This is the actual App that presents the GUI shown in Figure 1 and is operated by a gatekeeper at a consumer show to validate tickets for people entering the show.

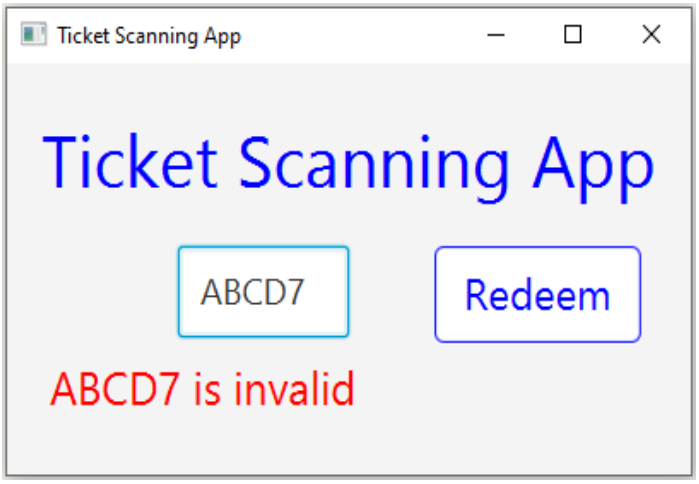


Figure 5. ABCD7 does not appear in codes.txt

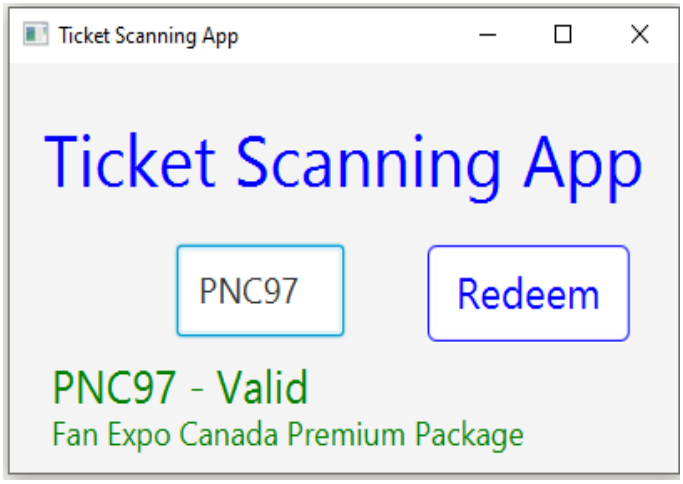


Figure 6. A valid code

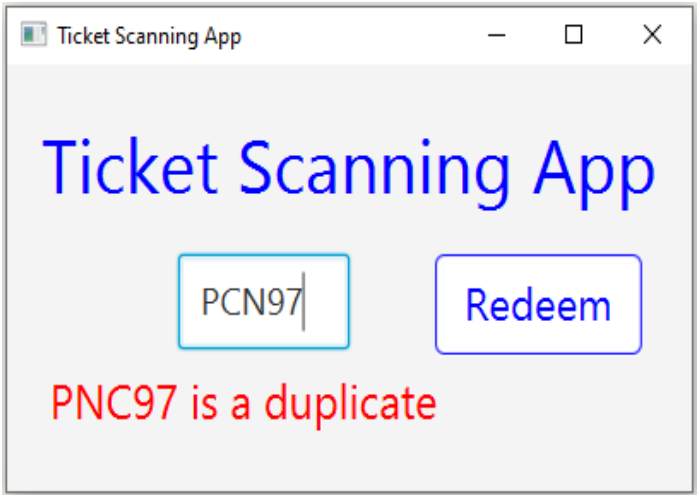


Figure 7. This person is trying to enter the show multiple times. PNC97 has previously been scanned.

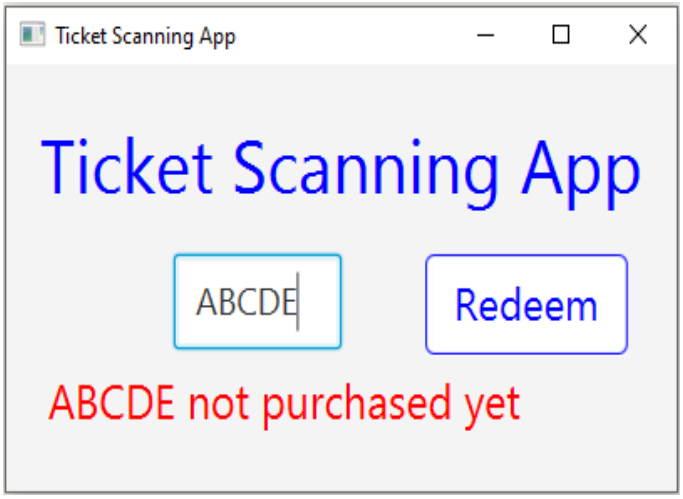


Figure 8. This code has not been bought yet.

Figure 9. A Symbol USB Scanner

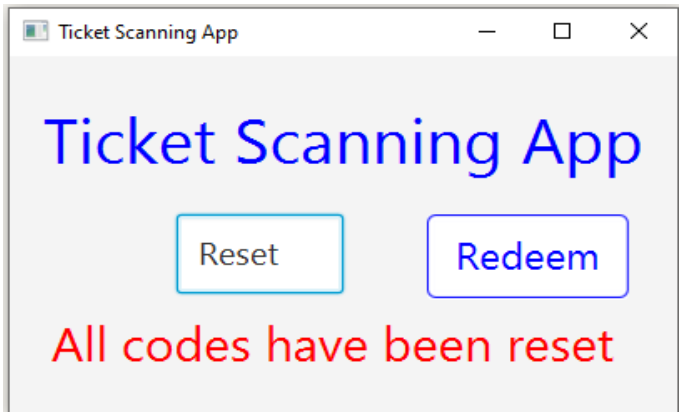


Figure 10. All codes are set to unredeemed

Note: **target** below refers to the value found in column 0; target in our ticketing case is a primary key

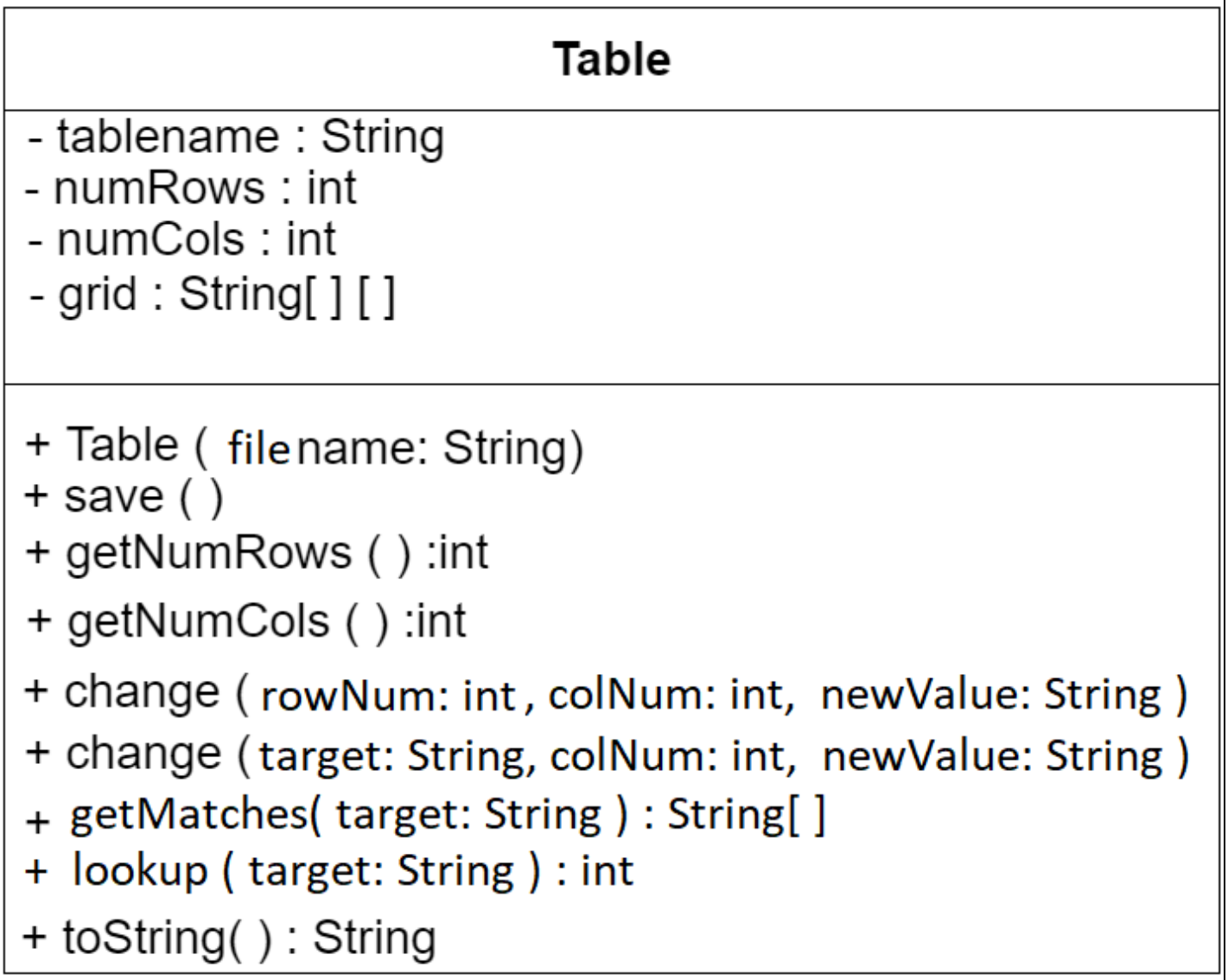


Figure 11. UML of the Table Class

About the Symbol USB Scanner

The scanner shown in Figure 9 is a USB device. The scanner acts identically to a keyboard. That is, if you open Notepad, and scan a barcode, you will see the barcode's text appear in Notepad window. The gatekeeper has the choice of either scanning a ticket with the scanner or hand typing in the barcode using the keyboard.

When the scanner scans a barcode, the alphanumeric characters are received as well as an ENTER. Because the scanner process does not end up pressing the REDEEM button, we need to act when the ENTER is encountered. See the java code at, "SENSING if an ENTER key has been Pressed while a TextField is in focus"

```
Enter the name of the tab delimited text file you wish to manage (e.g. codes.txt) > codes.txt
Successfully loaded: Table: codes.txt  rows = 119  cols = 4


Table Testing Menu

1. Display all data
2. Lookup
3. Search
4. Change
5. Save data to codes.txt
6. Get Single Cell Value
7. Save Single Cell Value
9. Quit
Select >
```

Figure 12. A TestTableProgram class menu

Extra Challenges ("…to make your App even better, try these challenges…")

ADD SOUND

```
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;

MediaPlayer player = new MediaPlayer
(new Media("file:///"+System.getProperty("user.dir").replace('\\','/').replaceAll(" ",
"%20")+"/"  +    "invalid.wav")  );
player.play();
```

STYLING a BUTTON

```
        myStyle =  "-fx-text-fill: blue;";
        myStyle += "-fx-background-color: white; ";
        myStyle += "-fx-border-radius: 5; ";
        myStyle += "-fx-font-size: 2em; ";
        myStyle += "-fx-border-color: blue";

        redeemButton.setStyle(myStyle);
        redeemButton.setMinWidth( 100 );
```

STYLING a TEXTFIELD

```
        input.relocate(100,100);
        input.setFont(new Font("System",20));
        input.setStyle("-fx-min-width: 50;-fx-min-height: 50;");
        input.setPrefWidth(100);
```

SETTING FOCUS

```
        input.setText("");
        input.requestFocus();
```

SPECIAL RESET CODE

Before the show begins the hardware and software need testing. Build a secret code, for example "RESET" which resets all tickets to "unredeemed".

SENSING if an ENTER key has been Pressed while a TextField is in focus

import javafx.scene.input.KeyCode;

```
        input.setOnKeyReleased(event -> {
            if (event.getCode() == KeyCode.ENTER){
                doRedeem();  //the routine that handles ticket validation
            }
        });
```

(place this code in your GUI file in section // 4. Configure the components)

COMPULSORY REQUIREMENTS

1. build a Table class and a Table test program with a menu like or similar to Figure 12 which provides options which test your Table class methods.

2. build the TicketScanningApp which presents the GUI in Figure 1 and can validate barcoded tickets.

OPTIONAL REQUIREMENTS

1. Make your app work with the USB Scanner shown in Figure 9.

2. Add two sounds, a valid.wav and invalid.wav sound.

3. Style your buttons and text.

4. Build a RESET code to set all tickets to unredeemed. This will allow the gatekeeper to test things before the show starts.

5. Use a Symbol Scanner as shown in Figure 9 to redeem tickets.
(see the section on 'Sensing an ENTER key pressed…."

6. Sort the codes.txt file by code, i.e. column 0 and then use Binary Search to locate codes.