Programming Graphical User Interfaces (GUI) with Java

2/2

Antonius Camara

About Event Handling (last week)

LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

- Event handling is a fundamental concept related to GUI's and how to process user interactions with the GUI (events)
- The slides provided by the teacher and the exercises performed in the contact sessions are not enough to ensure a comprehensive learning about the topic
- To properly learn the concept, you *have* to refer to a text book that comprehensively explains the concept
- The following book contains a very good coverage about the topic:
  - Java – How to Program, 8th Edition, Paul Deitel, Pearson, 2010 (Hard copies available from Laurea's library)
  - Chapter 14 (GUI Components: Part I)

How to implement Apps with many screens

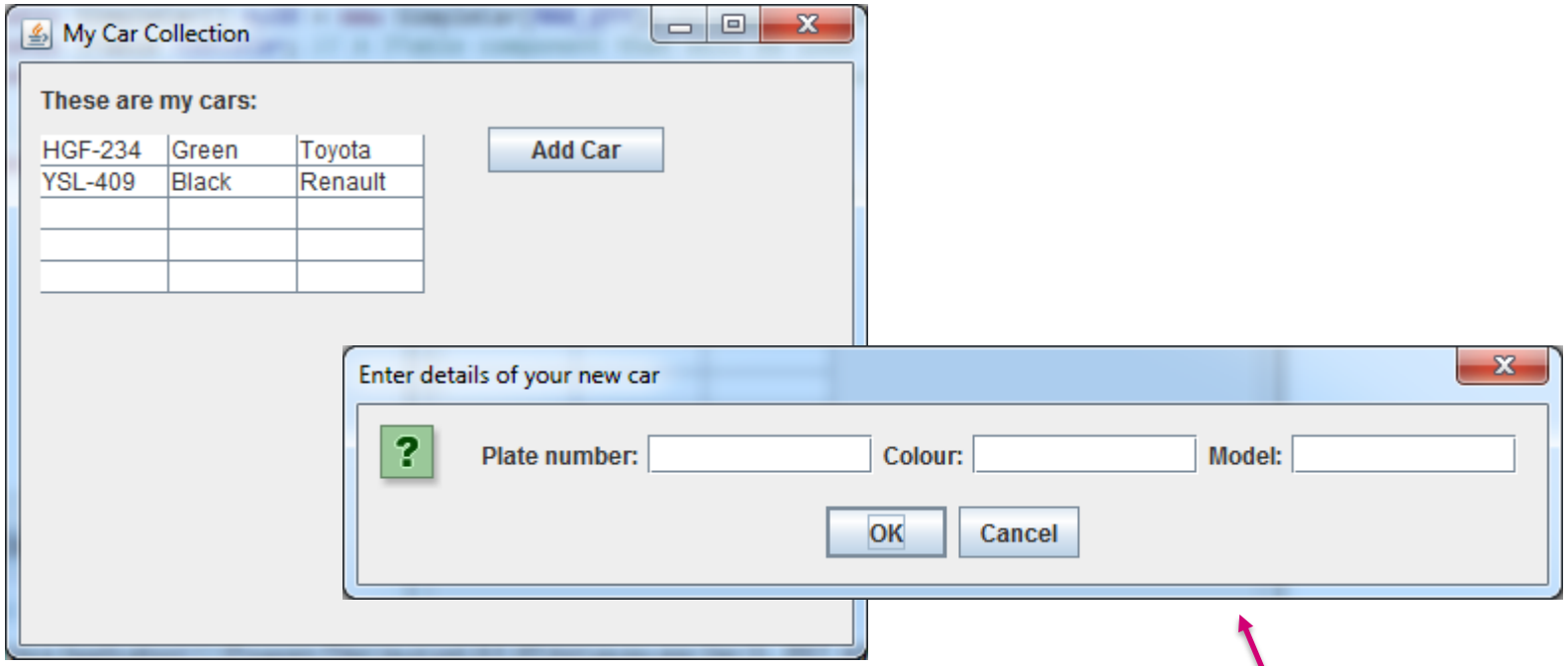
LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

- Having one "Main" screen and using dialog boxes to interact with the user
  - Standard dialogs (JOptionPane)
    - Messages (Errors, Alerts, Info), Confirmations, Input dialog, Option dialog
  - Custom dialogs (JDialog)
  - Beginning Java 8 API's, chapter 2: Swing components, sections → Custom dialogs, Standard dialogs
  - Applications with a more complex usage of screens: Beginning Java 8 API's, chapter 3 Advanced Swing, section → Multiple Document Interface Application
- Explore other options from online resources
  - Ex: Using CardLayout Layout Manager (http://stackoverflow.com/questions/27687427/how-to-create-a-swing-application-with-multiple-pages)

Swing Apps with many screens – In this course

LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

Having one "Main" screen in your app and using dialog boxes to interact with the user



Main screen

"Add Car" dialog box

Input dialog with several input fields

- Use JOptionPane.showConfirmDialog()
- Give as a second parameter a JPanel object containing the input text fields

```
JOptionPane.showConfirmDialog
(
    null,
    myPanel,
    "Enter details of your new car",
    JOptionPane.OK_CANCEL_OPTION
);
```

Example code

LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

```
// Defining the text input fields we will need in the dialog
JTextField plateNrField = new JTextField(10);
JTextField colourField = new JTextField(10);
JTextField modelField = new JTextField(10);

// A JPanel is needed so we can add UI components to a standard dialog
JPanel myPanel = new JPanel();

// Adding the PlateNR field to the JPanel
myPanel.add(new JLabel("Plate number:"));
myPanel.add(plateNrField);

// Adding the Colour field to the JPanel
myPanel.add(new JLabel("Colour:"));
myPanel.add(colourField);

// Adding the Model field to the JPanel
myPanel.add(new JLabel("Model:"));
myPanel.add(modelField);

// Displaying the input dialog: a standard Confirmation dialog showing the text fields from the JPanel myPanel
int result = JOptionPane.showConfirmDialog(null, myPanel, "Enter details of your new car", JOptionPane.OK_CANCEL_OPTION);
```

Hands-on: Understanding the MultipleScreens.java program:

LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

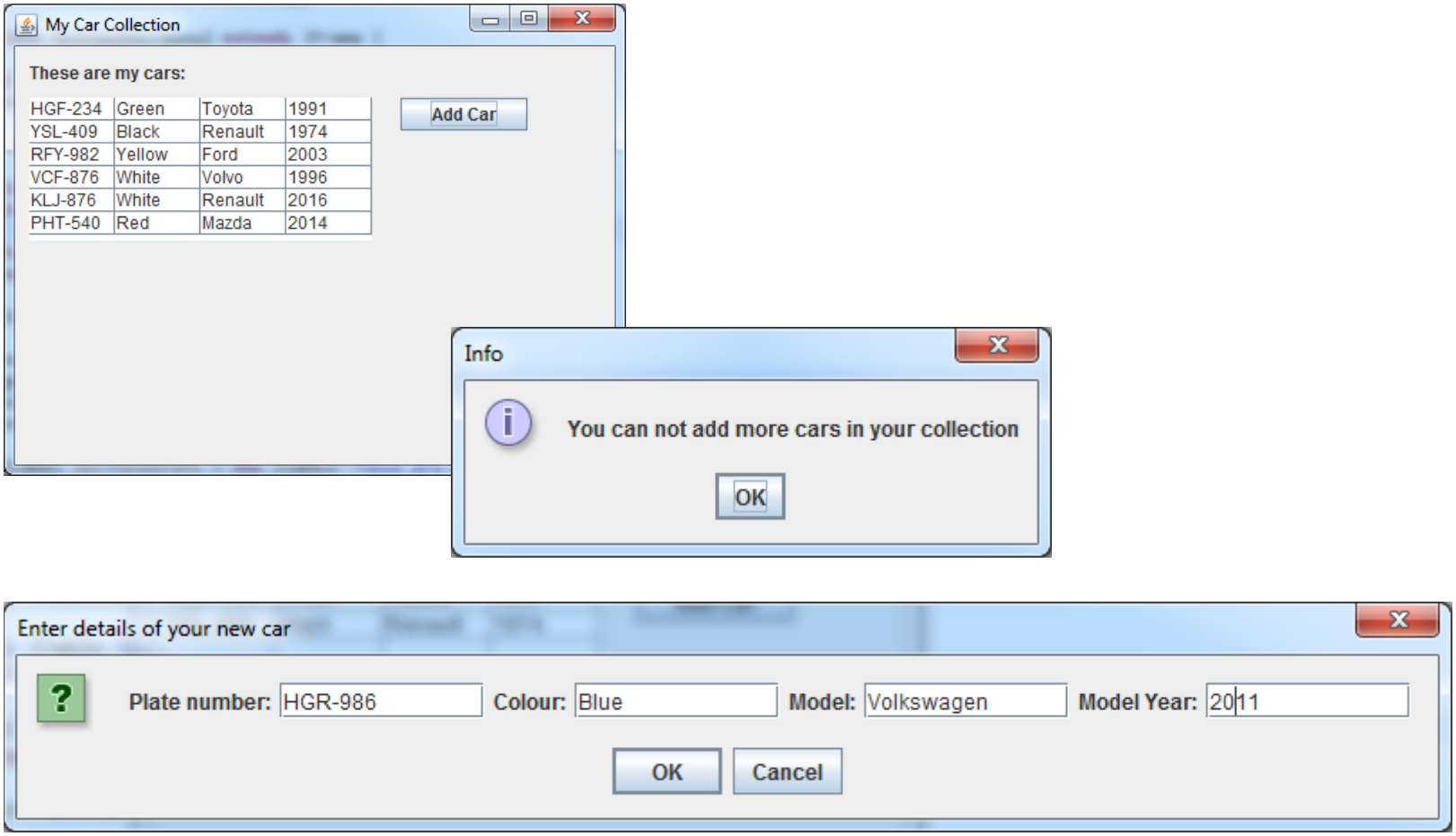
- Download from GitHub the files MultipleScreens.java and SimpleCar.java
- Create a project in Eclipse and add these two files (Classes) under the project's default package
- Test the application, inspect the code, and try to understand how it works. Answer the questions below to test if you have understood some basic concepts seen so far in the course:
  - What variables are declared as "Class variables" in the MultipleScreens.java file?
  - What java keyword is used to declare a Class variable?
  - Why the MAX_QTY variable is written in capital letters? What does the keyword "final" mean?
  - Why the array "myDB" is defined as a Class variable in this program?
  - What is the benefit of having the operations performed by "populateTable()" in an own, separate method?
- Try first to come up with your own answers. You can then compare your answers with the teacher's answers provided in the last slide of this presentation

Hands-on: Improving the MultipleScreens.java program:

LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

- Make the following changes in the program and test it to check if you got it right (check the outcomes in the next slide)
  - Increase the maximum amount of cars in the collection to 6 (MAX_QTY)
  - Add a new field to the Car object: “Model year”
  - Update items in myDB[] with this new field
  - Update the JTable tableCar to display the “Model year” column
  - The Add Car input dialog should also ask for this new field
  - When clicking on “Add Car”, if myDB[] already contains MAX_QTY items, the program should display a message dialog instead of the input dialog. The message displayed should be “You can not add more cars in your collection”, with an OK button

Hands-on: Improving the MultipleScreens.java program: Outcomes



Understanding the MultipleScreens.java program (Answers)

LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

1. What variables are declared as "Class variables" in the MultipleScreens.java file?
   Answer: MAX_QTY, dbItems, myDB, tableCar, btnAddCar

2. What java keyword is used to declare a Class variable?
   Answer: static

3. Why the MAX_QTY variable is written in capital letters? What does the keyword "final" mean?
   1. Answer: Variables written in capital letters are used to store CONSTANT values, i.e values that do not change during program execution. The "final" keyword is used to ensure that Java will allow the variable value to be set only once

4. Why the array "myDB" is defined as a Class variable in this program?
   Answer: Class variables can be accessed by any method and from any place within the Class where they are defined. In the case of myDB, we need to define it as a Class variable because we need to access and modify the content of myDB from the following methods: populateTable(), initiateCarCollection(), getNewCarFromUser()

5. What is the benefit of having the operations performed by "populateTable()" in an own, separate method – populateTable()?
   1. Answer: By doing this way, we are creating re-usable code. The operations performed by populateTable() can be used in different parts (by different method) of the application. For example, populateTable() is used in the constructor method MultipleScreens() and in the actionPerformed() method of the event handler class MyEventHandler