Seneca College

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

JAC444 **Submission date:** **March 24, 2023**

Workshop 6

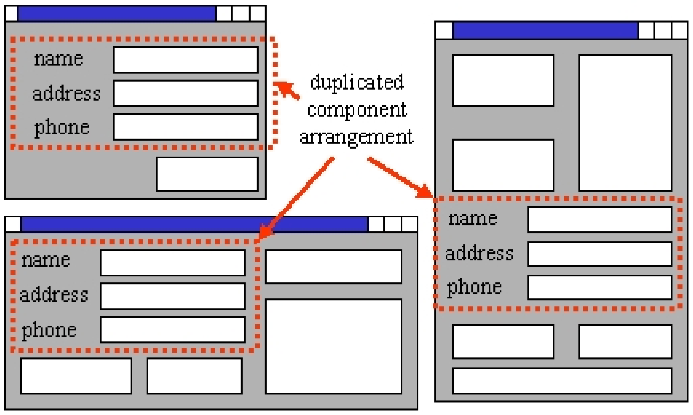
Description:

The following workshop lets you practice basic java coding techniques, creating classes, methods, using arrays, inheritance, polymorphism, Exceptional Handling, Java I/O, JavaFx, JavaFx Layouts, Containers and event handling

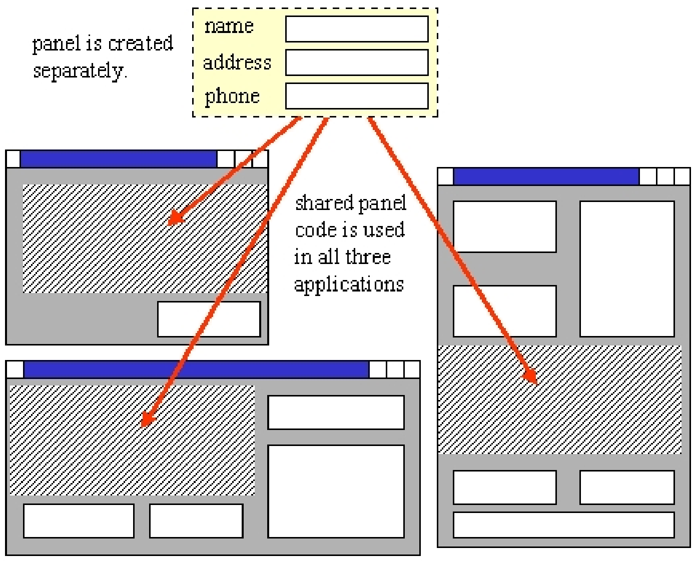
Task - 1

Grouping Components Together

It is a very good idea to keep your window components organized. It is often the case that an arrangement of components may be similar (or duplicated) within different windows.
For example, an application may require a name, address and phone number to be entered at different times in different windows →

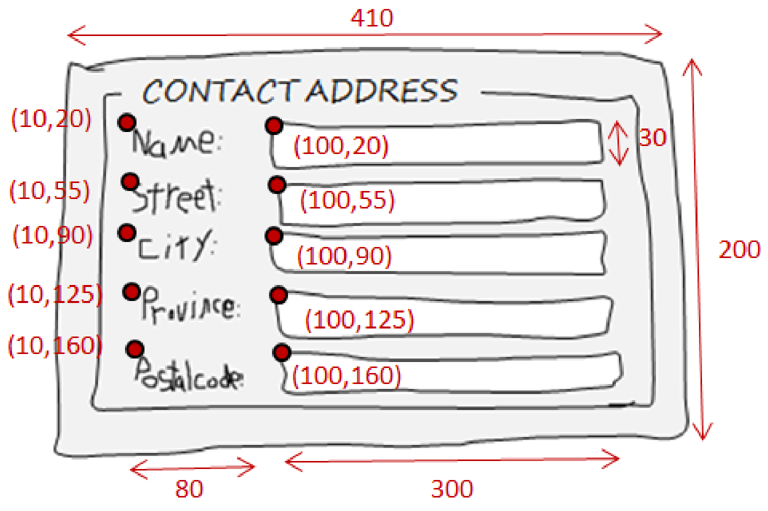


It is a good idea to share component layouts among the similar windows within an application so that the amount of code you write is reduced.
To do this, we often lay out components onto a **Pane** and then place the **Pane** on our window. We can place the pane on many different windows with one line of code … this can greatly reduce the amount of GUI code that you have to write.


panel is created separately.
name
address
phone
shared panel code is used in all three applications

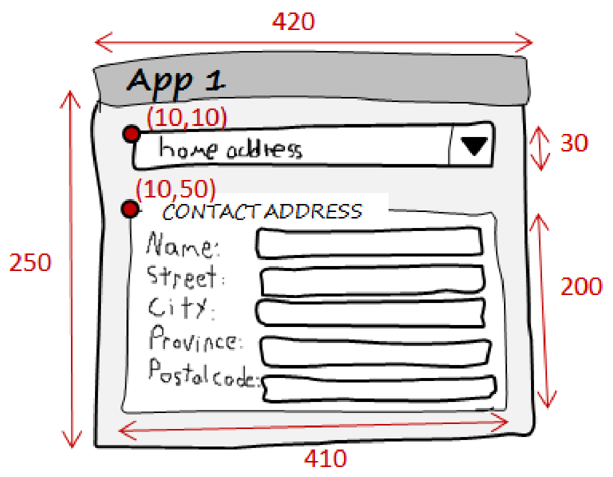
To do this, we often lay out components onto a **Pane** and then place the **Pane** on our window. We can place the pane on many different windows with one line of code ... this can greatly reduce the amount of GUI code that you have to write.
You will often want to create separate **Pane** objects to contain groups of components so that you can move them around (as a group) to different parts of a window or even be shared between different windows. The code to do this simply involves creating our **Pane** with its appropriate component arrangement and then adding the **Pane** to the window.
Pane objects are added to a window just like any other objects. So, we can have a **Pane** within another **Pane**.

Your **task** in which a **Pane** is used in more than one window. You will create a simple pane called **AddressPane** that contains 5 labels and 5 text fields for allowing the user to enter a name and address as shown below with its dimensions.



The Pane's size should automatically set according to the width and height of its components.

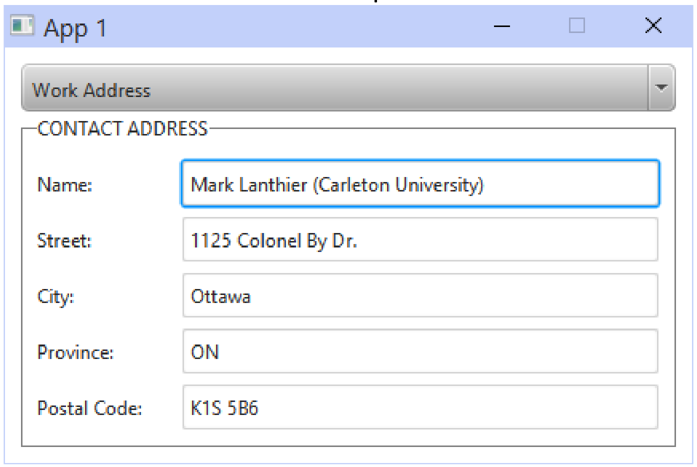
The **AddressPane** class itself is not a runable application (i.e., there is no **main** method). So, you should now make our **App1** application to test it out. Here are the dimensions for the application,



The **AddressPane** will now be "treated" as a single component and will be placed on the main window by specifying its location. We do not need to specify the size of the **AddressPane**, since this is fixed.

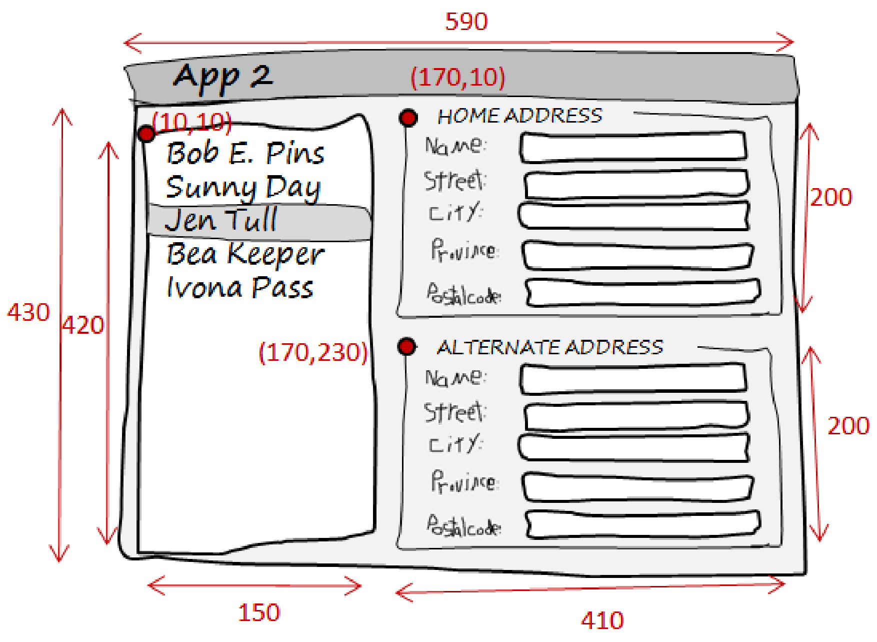
The topmost component here is called a **ComboBox** and it represents what is known as a drop-down list. It is similar to a **ListView**, except that only the selected item is shown and the remaining items can be shown by pressing the black arrow. More about **ComboBox** class functionalities details can be found here.

Here is what the application will look like when completed



Name: Mark Lanthier (Carleton University)
Street: 1125 Colonel By Dr.
City: Ottawa
Province: ON
Postal Code: K1S 5B6

Now let us consider a second application that makes use of the same **AddressPane** without altering the code in that class. Here are the dimensions for the 2nd application:



Once both the applications are designed properly you are supposed to update your second application **App 2** and make two buttons,

- First button is **Add** button upon clicking, the button should save the information and add the name in the ListView
- Second button **Remove** should be able to remove the contact from the ListView.
- Make sure you have proper checks on your application that if the user has not selected any name from the list view the remove button should be disabled.
- Before adding the details all the fields should be filled properly.

You are free to use either scene builder to build the app or just do it directly in the code. Follow proper coding design MVC as shown in this week lecture. Proper packages should be created and the files should in their proper packages.

Workshop Header

```
/*********************************************
Workshop #
Course:<subject type> - Semester
Last Name:<student last name>
First Name:<student first name>
ID:<student ID>
Section:<section name>
This assignment represents my own work in accordance with Seneca Academic Policy.
Signature
Date:<submission date>
*********************************************/
```

Code Submission Criteria:

Please note that you should have:

- Appropriate indentation.
- Proper file structure
- Follow java naming convention
- Document all the classes properly using JavaDoc
- JavaDoc should be generated properly in the project
- Do Not have any debug/ useless code and/ or files in the assignment

Deliverables and Important Notes:

All these deliverables are supposed to be uploaded on the blackboard once done.

- You are supposed to create **video with voice** of your running solution. **(50%)**
  - Screen Video captured file should state your last name and id, like Ali_123456.mp4 (or whatever the extension of the file is)

    OR

  - Show your work during the lab
- A text file which will reflect on learning of your concepts in this workshop. **(20%)**
  - Should state your Full name and Id on the top of the file and save the file with your last name and id, like Ali_123456.txt
- JavaDocs must be used for proper documentation of each task. **(15%)**
- Submission of working code. **(15%)**
  - Make sure your follow the "**Code Submission Criteria**" mentioned above.

    - You should zip your whole working project to a file named after your Last Name followed by the first 3 digits of your student ID. For example, **Ali123.**zip.
- Your marks will be deducted according to what is missing from the above-mentioned submission details.
- Late submissions would result in additional 10% penalties for each day or part of it.

Remember that you are encouraged to talk to each other, to the instructor, or to anyone else about any of the assignments, but the final solution may not be copied from any source.