Inf1B Joint OOP + D&A Assignment
Building a GUI

Hutchins, Nadarajan*

Out: March 2nd, 2007 Due: 5:00pm Friday March 23rd, 2007

Materials: Download the the following from the informatics 1 website:

`http://www.inf.ed.ac.ac.uk/teaching/courses/inf1/oop/assignments/assignment6.zip`

Submission: Type the following commands at the UNIX shell prompt to submit this assignment:

```
submit inf1 inf1b da6 da6.pdf                % D&A file
cd workspace/MyGUI
submit inf1 inf1b da6 DatabaseMain.java      % OOP file
```

The fine print: This project is due by 5:00pm on the due date. If your program does not compile, 30% will be deducted from your mark. Up to 10 points may be deducted for bad indentation. Late submissions will not be accepted. Collaboration with other students on this assignment is not permitted. If you have any questions on this assignment, please post them to the newsgroup `eduni.inf.course.inf1b` or ask a demonstrator in one of the drop-in labs sessions.

1 Introduction

You've seen widgets in a couple of labs, and the last OOP tutorial covered how to build a simple calculator. You have also covered structured data and SQL in the Data and Analysis portion of this course. In this assignment you will be applying knowledge that you've learned in both halves of the course. Your goal is to design and build a GUI that allows users to query a database.

There are two parts to this assignment – an OOP portion, and a D&A portion. Each part will be marked separately as a single assignment. For the D&A portion, you must design a GUI, according to the instructions on the D&A assignment sheet. For the OOP portion, you must implement your design in Java.

2 The OOP Portion of the Assignment

You can download the code for this assignment from
`http://www.inf.ed.ac.uk/teaching/courses/inf1/oop/assignments/assignment6.zip`

Create a new project named "MyGUI" within eclipse, and import the example code into it as usual. The code contains a simple widget library, along with a function that you

*Edited from an original version by Manuel Marques-Pita can use to query the IMDB database using SQL. To get the database stuff working, you will need to add the file `pg73jdbc3.jar` to your classpath. Do the following:

- Select "MyGUI" with the package explorer, and then click on "Project → Properties."
- Select "Java Build Path" in the left-hand column.
- Click on the "Libraries" tab.
- Click the <Add Jars> button.
- Select "pg73jdbc3.jar" from the list – it should be in the "MyGUI" folder.
- Click <Ok>.
- Click <Ok> again.

2.1 Example Code

The example code contains a GUI library called `widgets` which has been written especially for informatics 1. You do not (and should not) use the standard Java libraries for this assignment. Java comes with a very sophisticated GUI library called *Swing*, which is described in the textbook by Deitel and Deitel. Unfortunately, although Swing is a powerful library, it is also very complicated, and it takes a good deal of time and experience to master. The widgets library provided here is much smaller and simpler.

The full documentation for the widgets library can be found at:

`http://homepages.inf.ed.ac.uk/s0341095/GUIDoc/index.html`

An example program has been provided which demonstrates how to use the library. "CalculatorMain.java" is the same calculator program that you discussed in tutorial.

2.2 Widgets

The inheritance hierarchy for the widgets library is shown in figure 1. There are three kinds of widget, which are described below.

Simple widgets, such as `TextLabel` and `TextArea`, just draw something on the screen. They are used to display information to the user, but they do not respond to any events.

Controls, such as `Button` and `Checkbox`, generate actions. Whenever the user clicks on a button, clicks on a check box, or changes the text in a text field, it triggers the `doAction()` method for that widget. In order to build a GUI, you must create subclasses which override `doAction()`.

WidgetGroups, such as `VLayout` and `HLayout`, are not visible on the screen. A widget group packs group of smaller widgets into one big widget, and lays them out into neat rows and columns.

The widget classes are summarized below. For more information on how to use these widgets, please refer to the online documentation.

- A `TextLabel` contains a single line of text, which is usually a few words or a short sentence. The text cannot be edited by the user. As its name suggests, TextLabels are generally used to label various parts of the GUI so that the user understands what to do.
- A `Button` should be self-explanatory. It triggers `doAction()` whenever the user clicks on it.
- A `CheckBox` consists of a label, along with a small box which can be either checked or unchecked. It triggers `doAction()` whenever the user checks or unchecks the box.
- A `TextField` contains a single line of editable text, where the user can enter names, addresses, etc. It triggers `doAction()` whenever the user changes the text.
- A `NumberField` is a special kind of `TextField` for getting numeric input from the user. It will ensure that the user has entered a valid integer, and provides methods which convert the text to and from a integer.

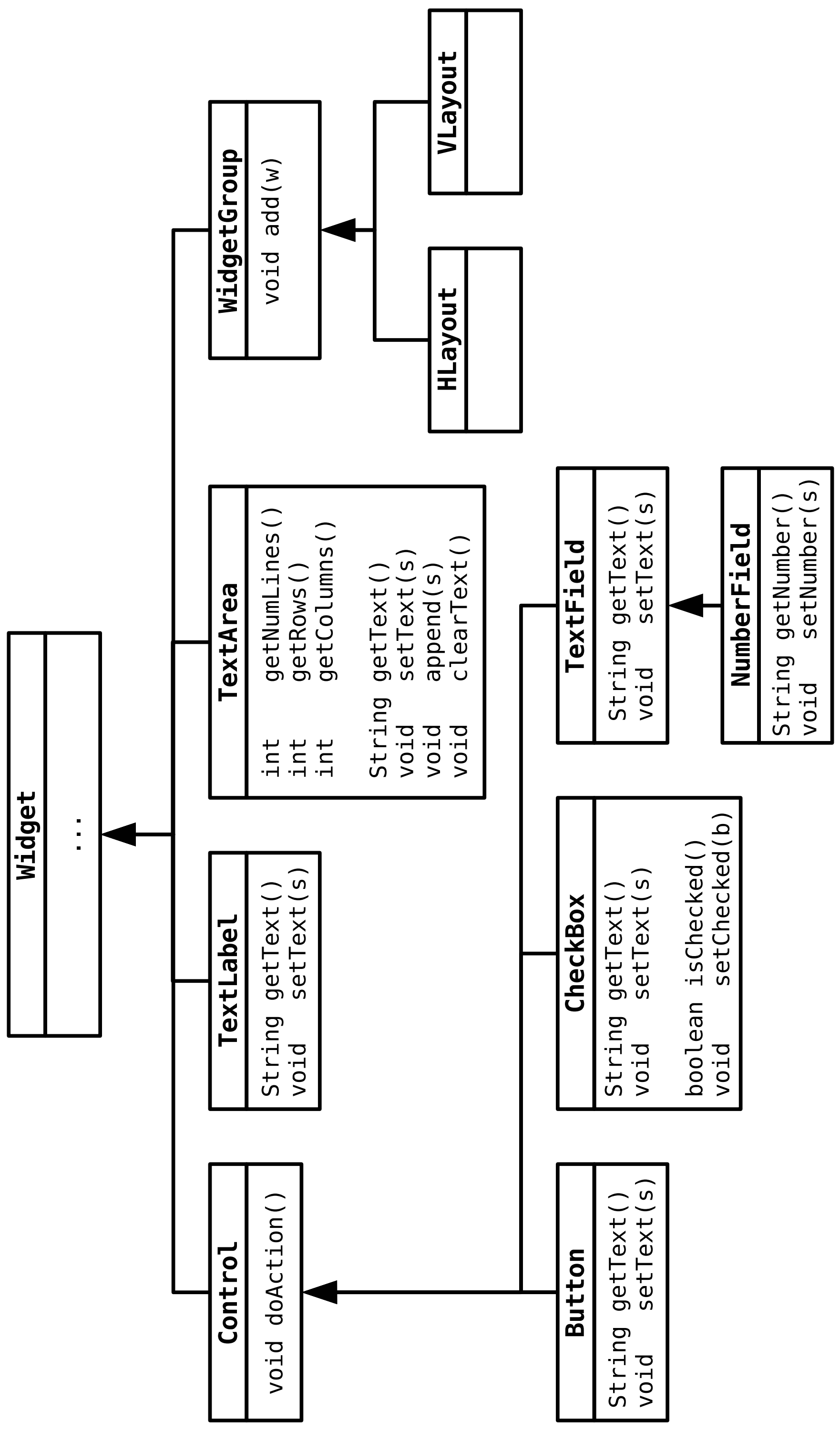


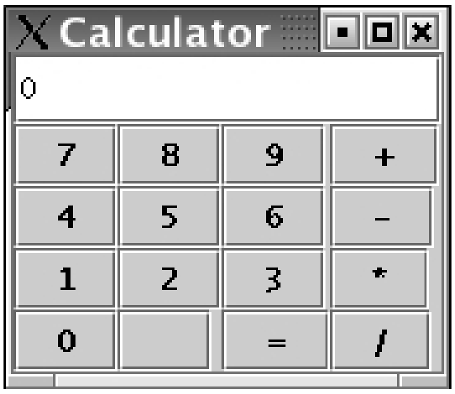
Figure 1: The Widget Hierarchy

- A `TextArea` is not editable, but it can display several lines of text to the user. It is useful for showing large amounts of output.

- An `HLayout` widget will create a horizontal row of widgets.
- A `VLayout` widget will create a vertical column of widgets.

2.3 Laying out the GUI

Every window on the screen has a single *root widget*, which is a `WidgetGroup` that contains all of the labels, buttons, text fields, etc. which are displayed within that window. There is no way to specify exactly where each widget should be placed within the window. The exact coordinates and size of each widget will vary depending on the screen resolution, font size, and the size of the window. Instead of specifying exact coordinates, you must organize your widgets into rows and columns. The example file "Calculator.java" shows how to lay out the following simple calculator:



Each group of four buttons is packed into a column using `VLayout`. The four columns are then packed into a grid using `HLayout`. And finally, the `TextField` at the top is placed above the buttons using `VLayout` again.

2.4 Building an Application

The main class for your application is found in "DatabaseMain.java". `DatabaseMain` inherits from `DatabaseApp`. `DatabaseApp` opens a new window on the screen, and connects to the inf1 database.

You must implement the `buildGUI()` method so that it constructs a *root widget* for your application. The `buildGUI()` method is triggered automatically by `DataBaseApp` when it creates the main application window.

In order to construct a GUI, you will also need to create a number of classes which inherit from the basic widgets, but which override `doAction()` to do different things. For example, "Calculator.java" defines three kinds of button:

- A `DigitButton` adds a new digit to the current number.

- An `OperatorButton` sets the operation to be performed.

- The `EqualsButton` performs the operation.

Each of these classes is defined *inside* the main application class. The main application class has fields which hold the current state of the application. The widgets you define will need to access and modify this state. (You will have to decide what the current state is, add the appropriate fields, and then initialize those fields in the constructor.)

3 Querying the database

The `runQuery()` method (defined in `DatabaseApp`) will query the database. It's defined as follows:

```
List<List<String>> runQuery(String sql)
```

`RunQuery` takes a single string as an argument, which should be a valid SQL statement of some kind. It will return a list of all the entries it finds. Each entry is a list of strings which contain the values requested by the SQL statement.

For example, the statement:

```
runQuery("SELECT title,year FROM movie WHERE year < '1920'")
```

will return a list of entries, where each entry is a list of two strings. The first string will contain the title of the movie, while the second contains the year. It is your job to take the list of lists of strings that `runQuery()` returns, and present them to the user in some sensible fashion.

3.1 Testing queries with psql

If you are having trouble getting your queries to work within Java, you may wish to test them first using the `psql` program that you used in an earlier D&A lab. To run `psql`, type the following at the command prompt:

```
psql -h pgteach inf1 inf1
```

The password is `inf1`. In addition to `select` statements, you may also find the following commands useful:

- `\dt` — get a list of all tables in the database.
- `\d` *tablename* — get a list of all columns in table *tablename*.

3.2 Lists and Iterators

Please refer to the lecture notes for more information on how to use lists in Java; what follows is a brief summary. The easiest way to process the list of entries is to use an extended for loop. The following example first queries the database to get a list of entries, and then iterates over the list.

```
List<List<String>> results = runQuery("SELECT title,year FROM movie");

for (List<String> entry : results) {
  /* ... do something with entry ... */
}
```

3.3 Working with strings

Another thing that you will have to do in this assignment is display textual information to the user. The '+' operator will not only concatenate two strings together, it will automatically convert other objects to strings. For example:

```
int    numCats = 23;
String mystr   = "My aunt has " + numCats + " cats.";
// mystr = "My aunt has 23 cats.";
```

4 The OOP portion of the assignment

For the OOP portion of this assignment, you must design and implement a new GUI. Your GUI should allow the user to request information by typing data into text fields, checking boxes, clicking buttons etc.

Once the user has requested some information, your program should generate an SQL `SELECT` statement which retrieves that information from the database. Use `runQuery()` to do the actual retrieval.

After your program has retrieved the information, it must format the results and present them to the user. The easiest way to do this is to format the results as a string, and then show that string within a `TextArea`. The string must be formatted so that it is readable!

4.1 Specifics

Download the example code from the website, and create a new project in eclipse as usual. The file you must submit for this assignment it called "DatabaseMain.java". For the most part, you are free to organize your code however you see fit. However, to help the markers mark your submission, you should structure your program according to the following basic outline.

- Implement `buildGUI()`. The `buildGUI()` method should return a single `HLayout` or `VLayout` object which contains all of the widgets in the GUI. The widget library will invoke `buildGUI()` automatically.
- Implement `buildSQLQuery()`. The `buildSQLQuery()` method should read whatever data the user has entered, and use it to construct and return an SQL query string. Note that `buildSQLQuery` is not called automatically; you will have to invoke it from within the GUI logic.
- Implement `showResults()`. The `showResults()` method should take the list of results from `runQuery()`, format those results in some way, and then show them to the user. Once again, `showResults` is not called automatically.
- Implement appropriate GUI logic. You will need to write some additional fields, methods, and widget classes which manage the internal state of the GUI in response to user actions. Place those above the `buildGUI()` method. Your GUI logic is responsible for calling `buildSQLQuery()`, `runQuery()`, and `showResults()` at the appropriate time.

4.2 Hints and tips

Hint: Start small. The first version of your GUI can be very simple. Start by writing a program which generates simple queries, and then shows the results. Once you get that working, you can start adding bells and whistles, one widget at a time. It is easy to get overwhelmed if you start writing a large GUI all at once.

5 The D&A portion of the assignment

5.1 Querying

The GUI developed should support the retrieval of information on the movie database that you have worked on in the first Data and Analysis Lab. Your application must enable users to perform the following queries:

1. What movies have been directed by a given **director**.
2. What movies have a particular **genre**.
3. What movies directed by a given **director** have ratings **higher than a given rating***.
4. What movies with a particular **genre** have votes **higher than a given vote***.

Note:* For the last two items (Queries 3 and 4), apart from the list of instances retrieved by the database, you also need to show a field in the interface showing the counts of instances for each of these two queries. The counting of the elements **must be done in Java and not using SQL queries. SQL queries must **only** be used to retrieve lists of results for a given query. Please consult Section 5.4 for the `SQL` queries that you must use to test your system.

5.2 Use Cases

You must design *Use Cases* to support your design implementation. Use cases essentially provide a basis for defining functional requirements of a system, in this case your movie querying and reporting system. The goal of doing this analysis is to get a very concrete picture of how the user (an actor) will interact with your system. In order to build your use case model you must use the conventions discussed in the lecture. Next year you will learn to convey the same information on a much simpler (UML) diagram. For the time being, however, the use cases need to be expanded in detail using structured text.

The main idea of this part of the coursework is that you identify a set of possible use cases relevant to the task of retrieving film information according to the queries listed above. You should use a *story board* approach during this analysis (as in the joke generation example illustrated by *Use Case 2* in lecture slides) so that you devise **one use case for each possible scenario** of interaction between the user and the tool you are designing. Add your use cases to the report that you will submit.

By doing this kind of analysis, you will be able to determine the features that your interface must have in order to be fully functional as well as usable by the users. In reality this process is done prior to code development in order to capture all the functional requirements which should be implemented. It would also help if you followed this practice in your assignment.

5.3 User Interface Design

Finally, provide a document for the choice of your interface design using **diagrams of detailed textual explanations** and a **detailed description of the interface** you will implement in Java. Make sure you define what components are used and justify the presence of each of them in the interface. A screen capture (or a drawing) of your GUI would be helpful for this part. Please refer to Section 1.1 of OOP Assignment 4 (Game Programming) on how to acquire a screenshot. You should also include some good **design principles** that you have adopted for your interface (e.g. Nielsen's Usability Heuristics). Provide justifications for the principles that you have identified.

5.4 Test Queries

In order to make your work easier, we will provide the SQL queries you will be using to test your application. Note that all of the provided queries start with a `SELECT` statement and end with a ";". In `SQL` these statements could be in uppercase or lowercase, however, actual database entries, such as director's name (enclosed in single quotes ' '), are case-sensitive. The whole query is actually one single line of text and should be treated in your program as one `String` object (for a given query).

1. In order to find the movies directed by a given director (for example Peter Jackson)

```
SELECT title FROM movie, moviedir, director
WHERE movie.mid = moviedir.mid
AND moviedir.did = director.did
AND director.fname = 'Peter (I)'
AND director.lname = 'Jackson';
```

2. In order to find the movies of a particular genre (e.g. Sci-Fi)

```
SELECT title, category FROM movie, moviegenre, genre
WHERE movie.mid = moviegenre.mid
AND moviegenre.gid = genre.gid
AND genre.category = 'Sci-Fi';
```

3. In order to find the movies that have been directed by a given director (e.g. Steven Spielberg) that have ratings higher than a given rating (e.g. 8.0)

```
SELECT title, rating FROM movie, moviedir, director
WHERE movie.mid = moviedir.mid
AND moviedir.did = director.did
AND director.fname ='Steven'
AND director.lname ='Spielberg'
AND movie.rating > 8.0;
```

4. In order to find the movies of a particular genre (e.g. Fantasy) and that have votes higher than a given vote (e.g. 40 000)

```
SELECT title, category, votes FROM movie, moviegenre, genre
WHERE movie.mid = moviegenre.mid
AND moviegenre.gid = genre.gid
AND genre.category = 'Fantasy'
AND movie.votes > 40000;
```

Important note. When testing your interface use **exclusively** the SQL queries provided in this handout and not others. The reason for this is that many of you will be querying the DB at the same time and queries producing long result lists will delay the system. This will cause delays in other student's queries and cause delays to you as well. Simply copying and pasting the above queries onto `psql` won't work due to the way single quotes copied from pdf files appear on the terminal screen. Hence you will need to type out part of the queries yourself, especially when a query involves single quotes, in which case all of them do.

Once again, you will need to think carefully about interface design issues here. Make sure that, as you think about possible use cases, you consider the set of interface components you need for your interface (remember that you only have access to those provided by the `Widgets` hierarchy).

In sum, for the D&A part of this coursework you need to produce and mark clearly:

1. A *Use Case specification.*
2. A detailed *Interface Design* document (diagram and/or textual form).

5.5 PDF file generation in Linux

For the D&A portion of the assignment you are required to submit a pdf file. In essence it does not matter how you come up with this file, but here are some tips for generating one in Linux. This could be achieved in several ways. A simple and straight-forward way is to use Open Office. Here are the steps:

1. Type `ooffice &` from your command prompt. Or select `Applications->Office->Word Processor`. This should invoke Open Office.
2. In Open Office, select `File->New->Text Document`. A blank page should appear where you can type your report.
3. To insert a diagram, select `Insert->Picture->From File` and select the picture from the directory where it is stored.
4. You can save your diagram in various formats, a suitable choice would be the Open Office format (sxw) which you will be able to edit later. To save it in pdf format, select `File->Export as PDF`. Choose a suitable directory to save your file, e.g. your home directory. Save it as `da6.pdf`. This is the final file that you will submit.
5. Acroread is a suitable program to view the pdf file. Type `acroread da6.pdf &` from your command prompt or select `Applications->Office->Adobe Reader` to invoke the program.
6. Please make sure that your **matriculation number** (and optionally **name**) appear at the start of your document.

Good Luck!