‘UMLator’

Our Solution to CS124 Group Project 2012

Connor Goddard (clg11), Sam Jackson (slj11) & Craig Heptinstall (crh13)

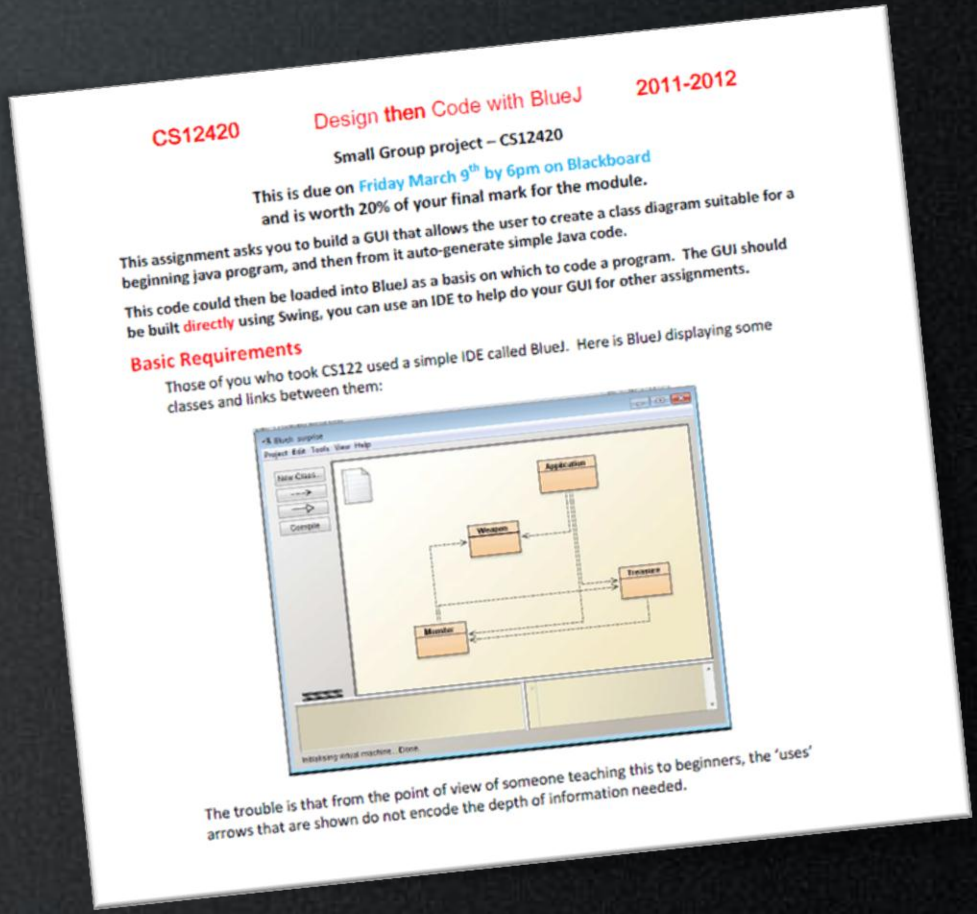
Analysis of the Task

Main Requirements:

- Draw UML class diagrams on screen (using interactive tools)
- Export those designs to Java code automatically
- Provide an intuitive and 'easy-to-use' UI

Other Requirements:

- Design GUI to replicate Blue-J interface
- 'WOW Factor' -:
  - Save/load projects
  - 'Undo' feature etc..



CS12420 Design **then** Code with BlueJ 2011-2012

Small Group project – CS12420

This is due on Friday March 9th by 6pm on Blackboard
and is worth 20% of your final mark for the module.

This assignment asks you to build a GUI that allows the user to create a class diagram suitable for a beginning java program, and then from it auto-generate simple Java code.

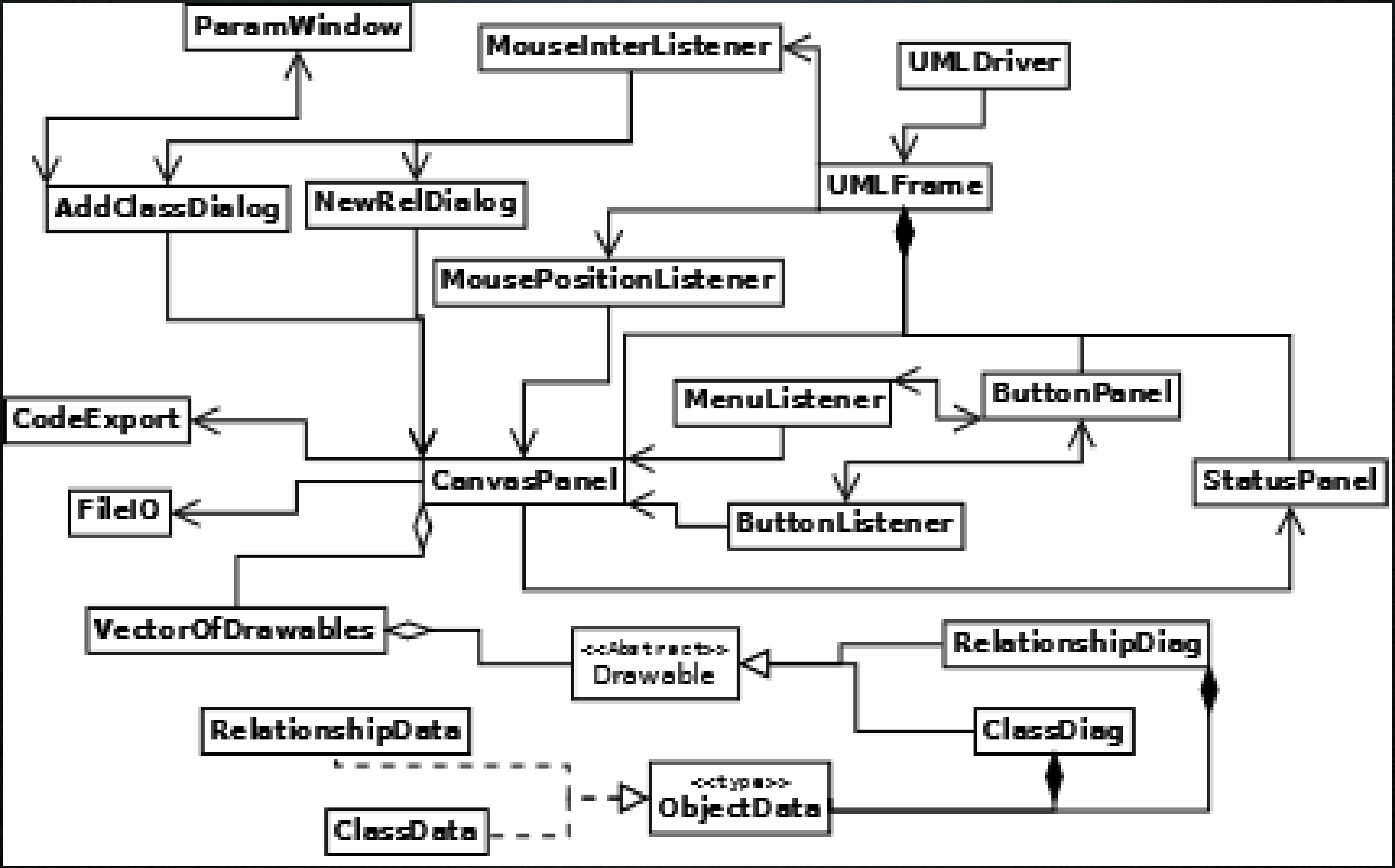
This code could then be loaded into BlueJ as a basis on which to code a program. The GUI should be built directly using Swing, you can use an IDE to help do your GUI for other assignments.

Basic Requirements

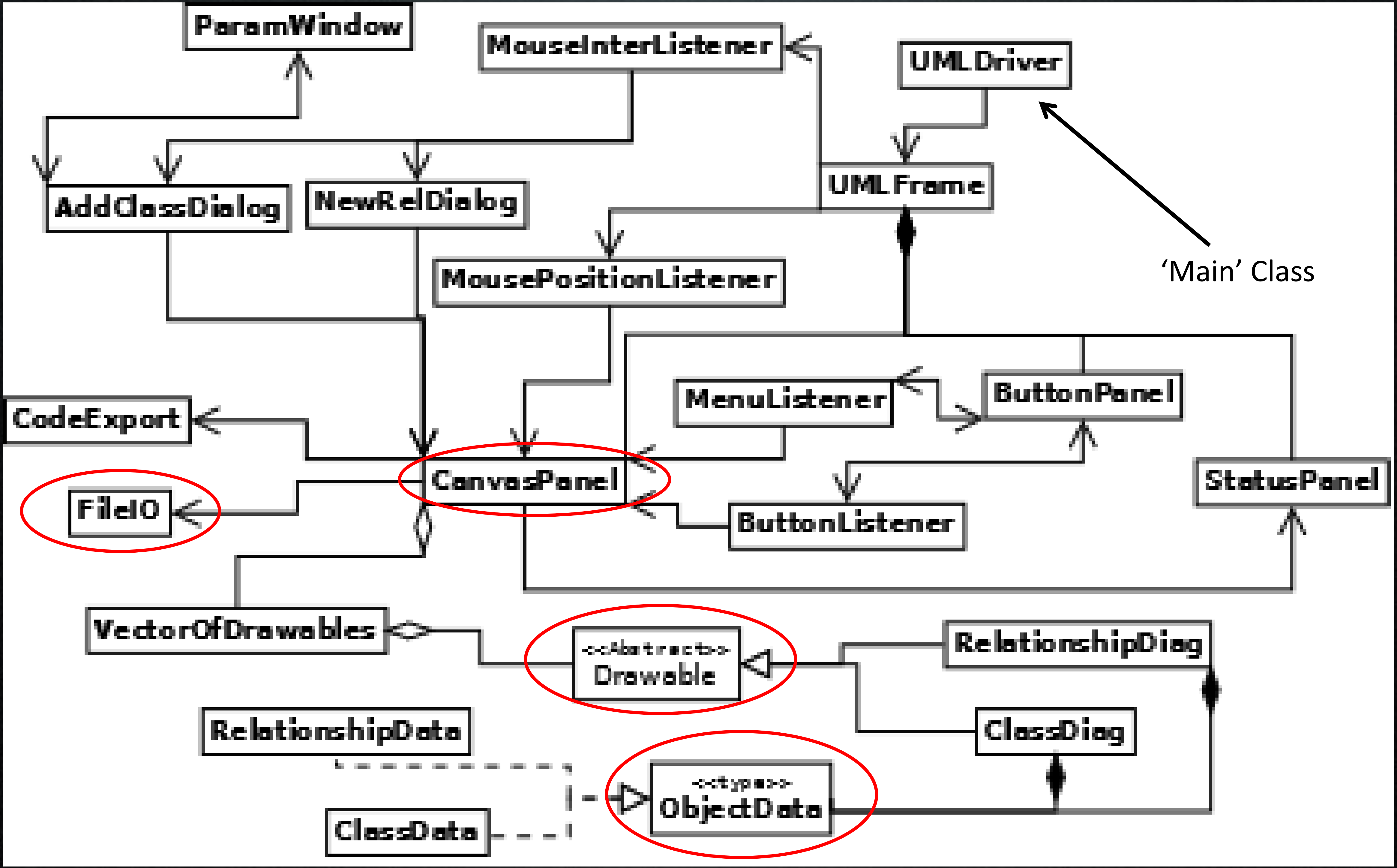
Those of you who took CS122 used a simple IDE called BlueJ. Here is BlueJ displaying some classes and links between them:

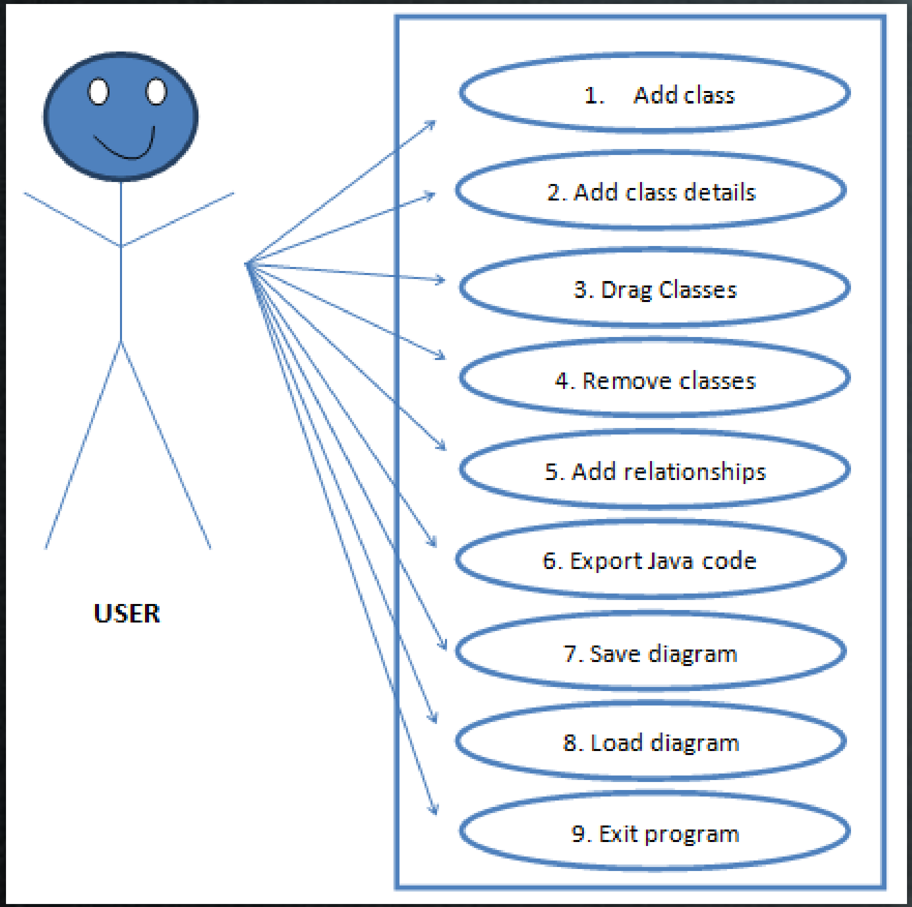
The trouble is that from the point of view of someone teaching this to beginners, the 'uses' arrows that are shown do not encode the depth of information needed.

Our Design

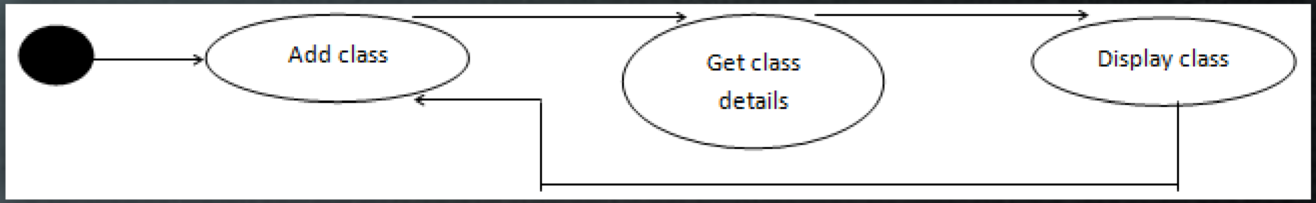

ParamWindow
MouseInterListener
UMLDriver
AddClassDialog
NewRelDialog
UMLFrame
MousePositionListener
MenuListener
ButtonPanel
CodeExport
CanvasPanel
FileIO
ButtonListener
StatusPanel
VectorOfDrawables
<<Abstract>>
Drawable
RelationshipDiag
ClassDiag
RelationshipData
<<type>>
ObjectData
ClassData

Our Design

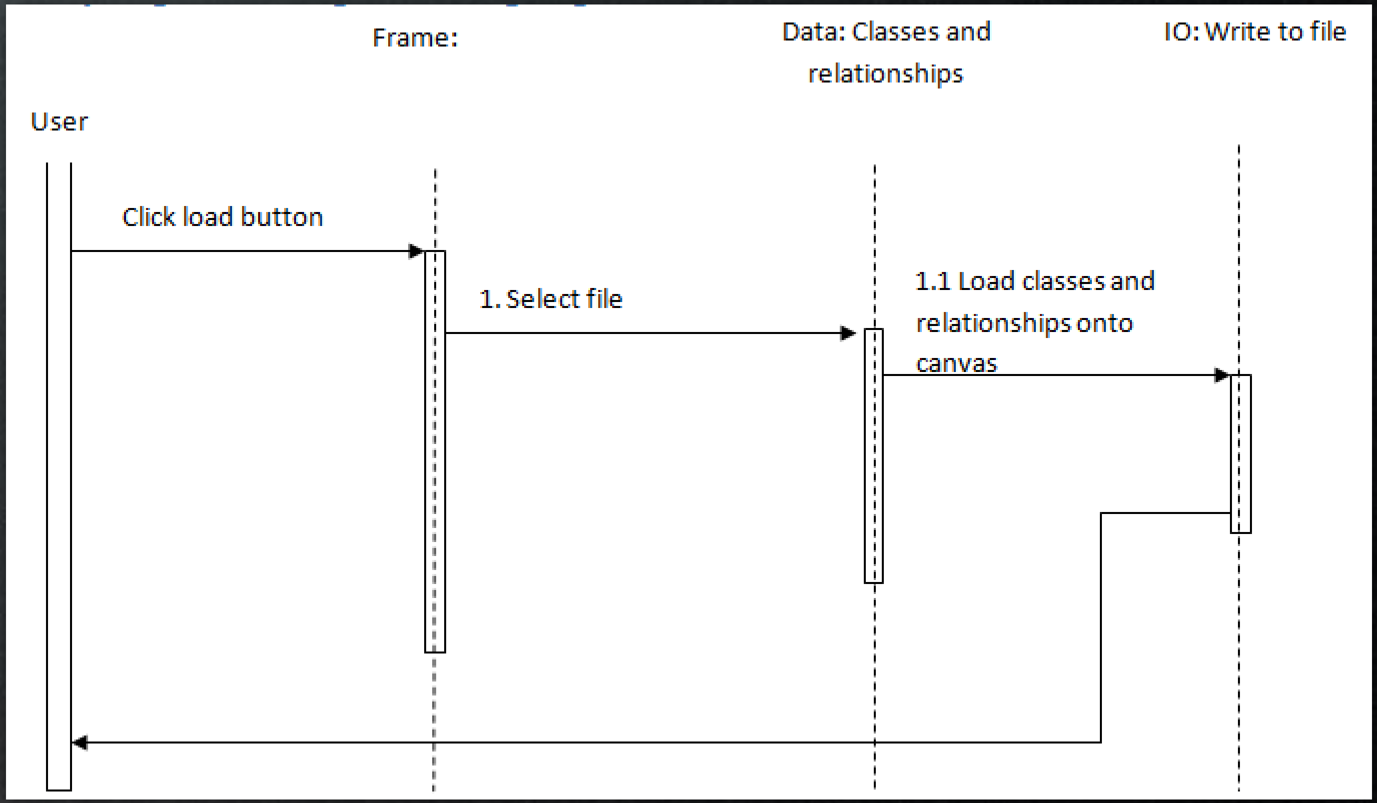

ParamWindow
MouseInterListener
UMLDriver
AddClassDialog
NewRelDialog
UMLFrame
MousePositionListener
'Main' Class
MenuListener
ButtonPanel
CodeExport
CanvasPanel
FileIO
ButtonListener
StatusPanel
VectorOfDrawables
<<Abstract>>
Drawable
RelationshipDiag
ClassDiag
RelationshipData
<<type>>
ObjectData
ClassData



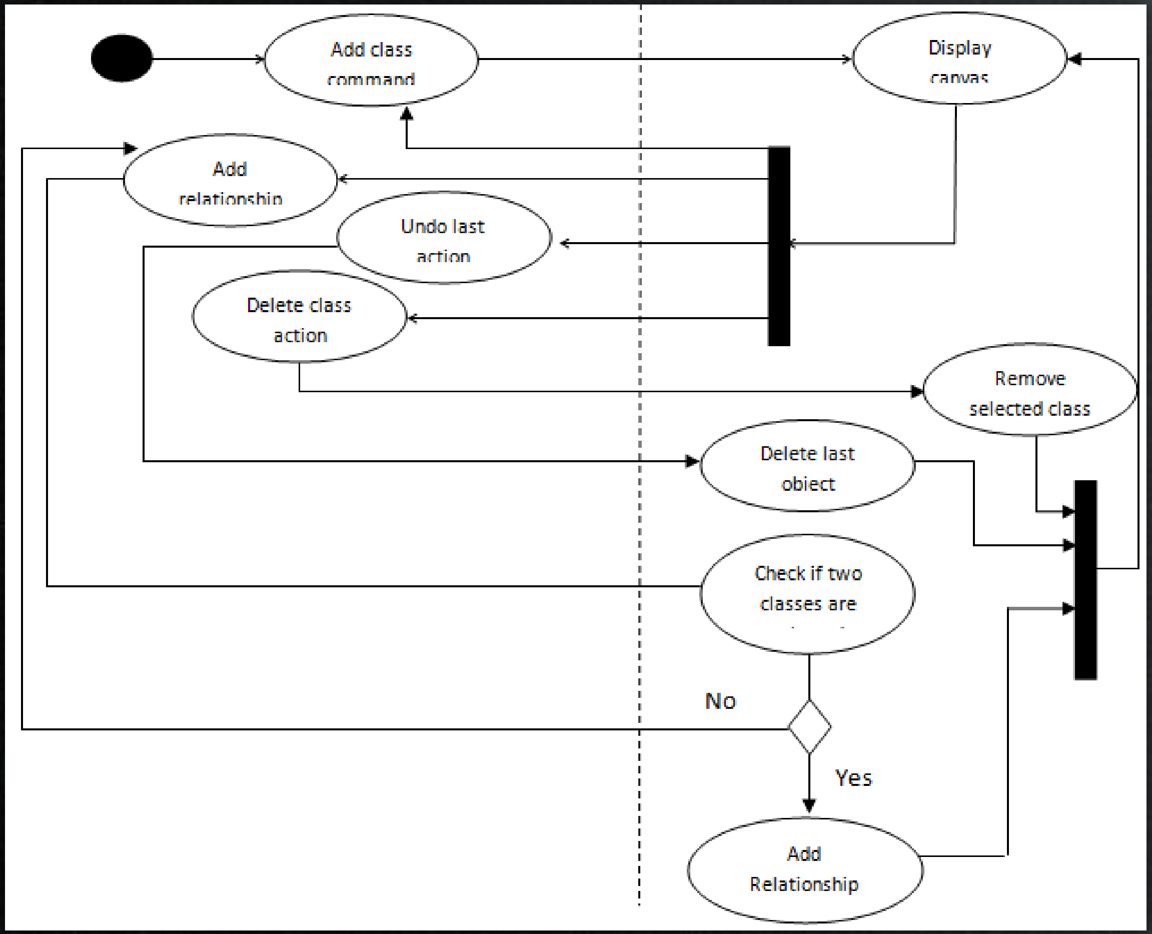
Use-Case Diagram



State Diagram


Frame:
Data: Classes and relationships
IO: Write to file
User
Click load button
1. Select file
1.1 Load classes and relationships onto canvas

Sequence Diagram



Activity Diagram

Tackling the Problem

Analysis & Design:

- ALL were responsible for deciding main solution
- Sam produced Class Diagram
- Craig produced other UML diagrams

Coding of Project:

Connor:

- GUI (except some dialogs)
- Line drawing algorithm
- ‘Linking’ all separately coded parts into one application
- ‘Undo’ feature

Sam:

- Data model for application
- Line drawing algorithm
- XML Import/Export

Craig:

- GUI dialogs

Documentation:

- Majority of documentation produced by Craig
  - (inc Justification of the Design, and Analysis & Design)
- Testing of application performed by Sam
- Project Evaluation / Self-Marking & Presentation produced by Connor

Tackling the Problem

However...

We all worked TOGETHER in ALL ASPECTS of the project..

..and helped EACH OTHER as required to ensure we produced a GOOD RESULT as a TEAM!

Evaluation

Overall, we were very happy with our team, and our final result:

Pro’s:

- Worked very well as a team
  - Good communication (inc. FB group)
  - Good time management
  - Dealt with problems efficiently/effectively
- Produced professional software:
  - Fulfilled all main requirements
  - Fulfilled other requirements
    - ‘WOW’ factors -> Save/Load etc..

Con’s:

- Hard to work on code – Risk of overwriting other’s code
- Could improve Analysis & Design
  - More time spent making sure ALL requirements are identified BEFORE producing designs/building
- JAR File Issues
  - Major problems with runnable JAR files
  - Could spend more time testing JAR files throughout project
- Minor XML Bug:
  - Detected bug that does not load instance variables from a saved XML project file – Solution found.

..and now for the DEMONSTRATION!

(fingers crossed)

Thank You.

Any Questions?