**OOP2 Hand-in Project Assignment, Semester 1, 2015**.

CA 5% first exam+10% Labs, **15% project**

**Due date** :

Project must be demonstrated to lecturer week beginning 7st December. You should have all of your documentation available at your presentation.

**Project Specification** (minimum requirements unless agreed with Lecturer):

* At least three classes required (at least 2 of which are instantiable)
* At least one of the classes is a GUI class, having a menu system plus at least three other ‘J’ components
* Each instantiable class should model a different aspect of a system (for

example a Book and a Student, as part of a Library system).

* The application should contain a ‘substantial’ processing element (an algorithm for doing something!). At a minimum, your system should allow the user to process at least one transaction involving an object from each class (eg process the return of a particular Book by a particular Student).
* Demonstrate superclass/subclass inheritance and/or composition/aggregation
* System should allow you to add, display and hold in memory in an appropriate data structure (yet to be covered) instances of each instantiable class,
* Save the data structure to file (yet to be covered) and load it up again.
* At least one class should be fully commented with Javadoc comments

**Note:**

Classes used as examples in class or featuring on lab sheets are not acceptable but can be used to form the basis of your classes. Each student should choose a different topic. All choices subject to lecturer approval.

**Documentation to be handed in:**

Requirement Specification (word), more than one paragraph, less than one page

Class diagrams for all classes (word), done up in Visio

Outline VOPC diagram for the GUI application (word)

Program code (.java)

Javadoc output for the javadoc commented class

You will also be asked to give a quick demonstration of your working project on the week beginning 7th December, **No demo = no marks.**

**Indicative marking scheme: Projects will be graded using the following criteria**:

* Scope/Complexity- safe, narrow, broad. (30%)
* Code features (classes, methods, structure choices, inheritance/composition, going beyond what we have covered) (30%)
* Quality of code (e.g. naming conventions, comments, indentation, methods, usability) (20%)
* Presentation (10 %)
* Documentation (10%)