**COEN 275 OO Analysis, Design and Programming Spring 2016**

**Assign 2 (175 pts) Due: 3rd May (10 pm)**

In this assignment, you will learn to

* Use Composition and Inheritance to achieve reuse
* Implement a design (in Java) that uses multiple inheritance

**Question 1 (125)**

This question tests your understanding of the concepts of inheritance and containment (as techniques of reuse) as they are applied to common artifacts that we are familiar with. In addition, the question allows you to be creative in applying the concepts to create additional artifacts of interest.

1. Firstly, you are required to define a few object types (classes) to represent a number of basic concepts: A **Rectangle**, a **Ball**, a **Picture**, a **Frame** and a **Block.**

Some of the important properties of the basic classes for the above concepts are given below. You are free to add any other properties that you may find applicable. Please note that not all properties may need to be stored as data members; some may be computed in methods.

**Rectangle**

*Dimensions (Length, Width), Area*

**Ball**

*Dimensions (Radius), Volume*

**Picture**

*Dimensions (Length, Width), Title, Artist*

**Frame**

*Dimensions (Length, Width), Area, Material*

**Block**

*Dimensions* **(***Height, Depth, Length)*

**Note**: You are free to define any other basic classes (similar to the ones above) of your choice.

**(50 pts)**

1. After you define the above basic classes, you must create some new classes (at least 5 new classes) via inheritance and composition of the basic classes above. For example, some of the new classes can be a **Square**, a **Box**, a **FramedPicture** and so on.

Points:

* You are required to define at least 5 new classes (using the basic classes via containment and inheritance) **(50 points)**

For each of the new classes you create (from the basic classes), define the following:

* Data Members
* Constructors
* Setters and getters (as applicable)
* A *toString()* method to show the contents of the object
* Other methods relevant to that particular class.

**Note**: *Make sure that some of the basic constraints (for example, the dimensions of a Frame object should be able to accommodate a Picture object, in a FramedPicture class)*

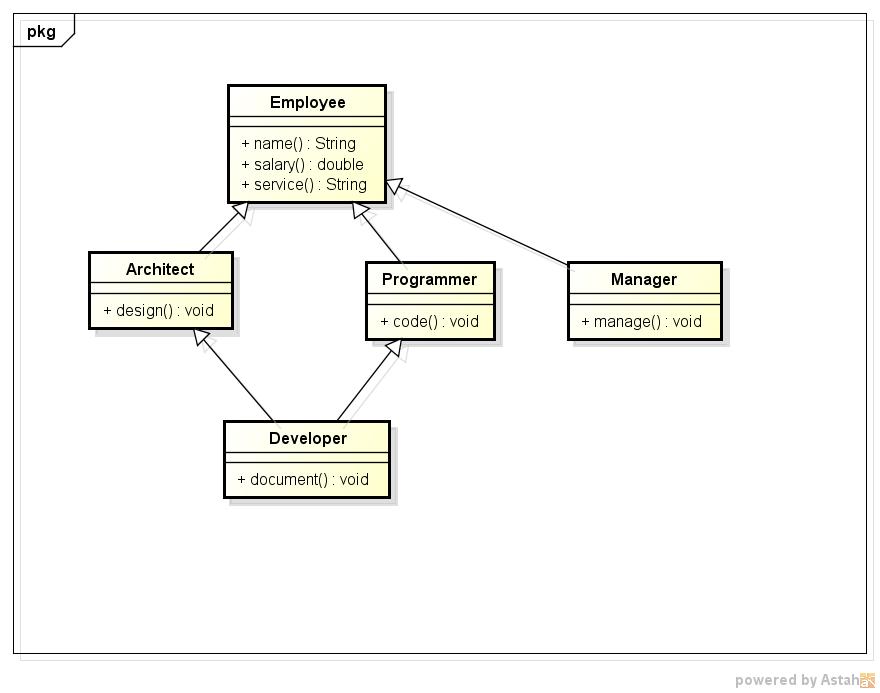
1. Define a **Question1Tester** class with main(), where you create instances of the classes you have defined and call the methods to show they work correctly. **(10 pts)**
2. Draw an UML class diagram (using an UML tool) showing the classes and relationships (containment and inheritance) among them. **(10 pts)**
3. Create Javadoc

**Point distribution:**

1. 50 pts
2. 50 pts
3. 15 pts
4. 10 pts

**Question 2 (50 pts)**

In this question, you will convert a design that uses multiple-inheritance (shown in the UML class diagram below), into a design that supports only single inheritance.



1. Draw an UML class diagram (using an UML tool) showing your re-design using single inheritance.
2. Implement your design – define the classes with the data members and methods shown in the diagram as well as any additional methods described below and others needed. For the method bodies, include a print statement, identifying the method name. Please note that the given design shows classes (not interfaces). So in your re-design, you must implement them as classes. Note: You are free to add any interfaces you may feel will enhance the design.

Please follow the requirements given below for each of the classes. You are free to implement other data members and methods you may deem necessary.

Class Employee:

* You are free to define the signatures of *salary()* and *service().*

Class Architect:

* The **design()** method should return a string “Design from “ with architect’s name.
* The **report ()** (not shown in the class diagram) should call **design()** and return the string returned by design().

Class Programmer:

* The **code()** method should return a string “Code from “ with programmer’s name.
* The **report ()** (not shown in the class diagram) should call **code()** and return the string returned by code().

Class Developer:

* The develop() method should return a string, that is a concatenation of the returned value from calling code() and design() methods. In the current design, these two methods are inherited from the two super classes, Architect and Programmer. In your re-design, you must make it possible for the developer to call these methods.
* The **report ()** (not shown in the class diagram) should call **develop()** and return r the string concatenated with the current Date.

**Class Manager:**

A Manager manages (in charge of a number of) architects, programmers and developers. A Manager class should have a data member (for example, a data member called **techList**) to hold instances of Architects, Programmers and Developers (not all Employees).

* The **manage()** should be able to call add and remove an instance of an Architect, Programmer or a Developer from techList.
* The **report()** of Manager should call report() of Architects, Programmers and Developers, in the techList.

**What to submit:**

Create a Jar file (see the instructions below), name it **assign2**\_***yourfirstinitialLastName*.jar and submit it on Camino.**

**How to create a Jar file in Eclipse**

Java allows you to group and package a number of files into one archive file (JAR file).

In Eclipse, you can create, export your project files (source code and other resources) into a Jar file and import a Jar file and extract the contents.

**To export a project to a JAR file**

1.    Start Eclipse and navigate to your workspace.

2.    In Package Explorer, left-click on the project you want to export.

3.    Right-click on the same project and select **Export.**

4.    In the Export dialog box, go to **Java** and click on **JAR file.** Click **Next**.

5.    In the Export dialog, select to export generated class files, all output folders and Java source files. Browse to select a location and create a name for your JAR file.

6. Click Finish. Your JAR file will be created in the location you selected.

**To import a project from a JAR file**

1.    Start Eclipse and navigate to your workspace.

2.    Create a new project and select a name for the project.

3.    Right-click on the **src** folder under the project name and select **Import.**

4. In the Import dialog, expand **General** and select **Archive File.** Click **Next**.

5.    Browse for the location of the JAR file that you want to import and click **Open**.

6.    The Import dialog box should show what’s in your JAR file.