Semester Project – Object Oriented Analysis and Design – Iteration 2

Before getting started on Iteration 2, be sure to carefully review the Project Introduction document, Iteration 1 document, coding demos (particularly Decorator and State) and get to know the JSON demos. The overall project is 40% of your semester grade and Iteration 2 is worth 20% of the project.

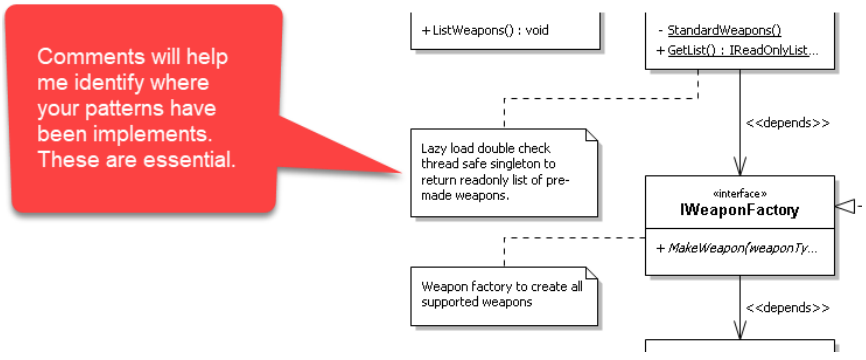
Iteration 2 Reminder (from Intro)

During iteration 1, you learned a lot about patterns during lecture. Next, you will update your class design to include the patterns you will implement. You will produce a very detailed class design that shows your pattern implementations and non-domain features such as persistence. Once again, you write a one-page document describing your designs to convince the CTO you are ready to go the next iteration. To get full payment for this iteration, the CTO requires your work to demonstrate that you are going to build something that exhibits excellent OO design principles, including SOLID, GRASP, and implements the GoF patterns he requires.

Iteration 2 Grading Rubric

1. **Updated Class diagram: 75%**
   Using the class diagram you completed in Iteration 1, add the **five** patterns described below. The patterns must show the interfaces, abstract classes, and concrete classes your application will implement to achieve the requirement. Show the appropriate relationships, including inheritance, realizations, and dependencies.

   Also include at least one comment per pattern pointing out where to find it in your diagram (3 comments for the singleton part).



   a. **Pattern 1: Strategy – 15%**
      You will use the Strategy pattern to implement reading and writing Trip objects to

disk using JSON or XML serialization.

When you write a trip to disk, it will store its current state and all of the details entered via the state machine. When you reload a Trip from disk, you should be able to drop it back into the state machine (if not complete) and resume work on finishing the Trip. If it is complete, it will be ready to generate an itinerary.

Diagram the strategy pattern for the JSON and XML persistence. Show the pattern for both read and write functionality. Note that if you are a team of 1, you must still design the XML functionality, but you do not have to implement it when coding the project. Teams of 2 will both design and implement both JSON and XML functionality.

b. **Pattern 2: 2 Factories – 15%**
   You will use the Factory pattern to make the persistence objects used for reading and writing Trip objects to/from using JSON or XML serialization.

   i. Design a Read factory to return the concrete persistence objects you designed in the Strategy pattern. Note that if you are a team of 1, you must still design the XML functionality, but you do not have to implement it when coding the project. Teams of 2 will both design and implement both JSON and XML functionality. – 7.5%

   ii. Design a Write factory to return the concrete persistence objects you designed in the Strategy pattern. Note that if you are a team of 1, you must still design the XML functionality, but you do not have to implement it when coding the project. Teams of 2 will both design and implement both JSON and XML functionality. – 7.5%

c. **Pattern 3: Decorator – 15%**
   You will use the Decorator pattern to implement the itinerary shown in the Project Introduction.

   Diagram the decorator pattern for the Itinerary. Carefully review the itinerary from the Project Introduction and design the components needed to make each section. Remember to keep to the single responsibility principle. The Lecture 14 and 15 coding demos and slides nicely demonstrate the how to implement an

itinerary using decorator.

d. **Pattern 4: State – 15%**
You will use the State pattern to implement the workflow to build a trip from start to finish. If done properly, your State pattern will enable you to return to incomplete trips at any point and resume your work. Once the pattern reaches the complete state, the trip should be valid and 100% ready to generate an itinerary.

Diagram the state pattern to handle creating a single Trip. Carefully review the state diagram from Iteration 1 and review the Lecture 15 slides and coding demos.

e. **Pattern 5: Three Singletons – 15%**
You will use the Singleton pattern to preload the reference data used by your application.

i. Package singleton – 5%
Design a singleton to load all packages into a singleton when your application starts. The agent will use this singleton when choosing the packages to add to a Trip.

ii. Person singleton – 5%
Design a singleton to load all people into a singleton when your application starts. The agent will use this singleton when choosing people to add to a Trip.

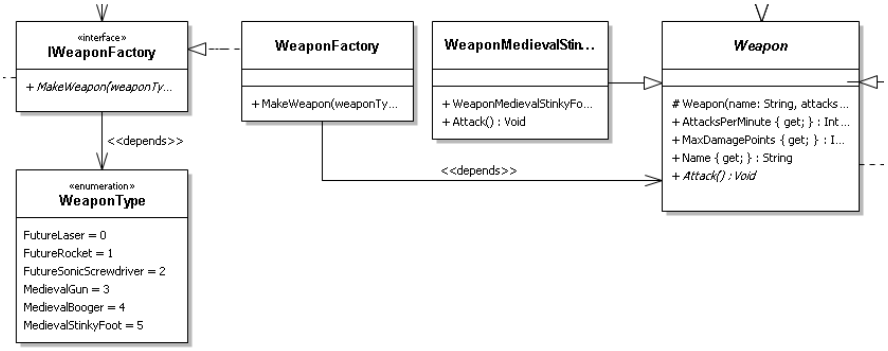
iii. Travel agent singleton – 5%
Design a singleton to load all TravelAgents into a singleton when your application starts. The agent will use this singleton when choosing himself/herself when the application starts. Think of this as a login function, though we are not bothering with passwords and usernames – the user will just choose an agent from a list in the singleton.

Your diagram will be graded for accuracy, completeness, and level of professionalism (so if it's messy, expect that to count off). I suggest

including comments to point out important facets of your work.

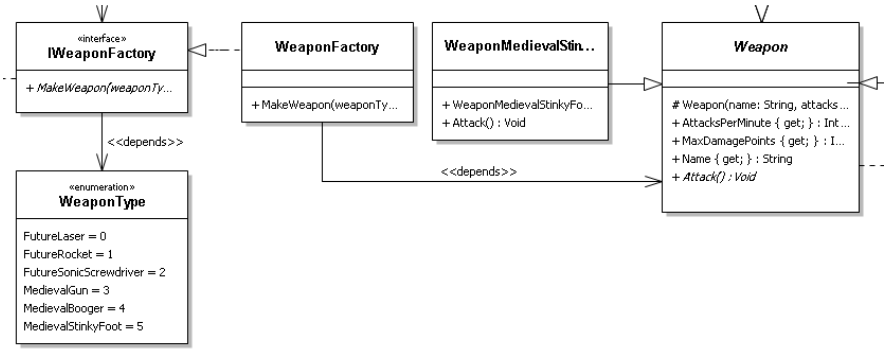
2. **Short writeup: 25%**
   Produce a Word document that has screenshots of the specific parts of your Iteration 2 class design document showing each pattern implemented in the prior section, along with any details you feel I need to understand. Be as brief as possible. Your document will look like this:

   a. **ReadPersistence Factory**



   Whatever notes you need to include to explain your work.

   b. **WritePersistence Factory**



   Whatever notes you need to include to explain your work.

   c. *etc*. There will be **eight** screenshots – one for each pattern from the prior section.

   I expect professional language, **zero** spelling and grammar issues, and a brief, informative summary of the work. The CTO at Premium is very busy and doesn't want to read a bunch of dull, obvious stuff, such as what the definition of each diagram is… he already knows that. Tell him something significant. Point out risks and offer solutions. Be a great consultant so you look good and so does the CTO. That's how you get more work and build a great reputation.

3. **Submission files – penalty of no less than 25% if not followed.**
   You must submit 2 files zipped into a single file. The filename will be **YourName_Iteration2.zip**. If you are a team of 2, the filename will be **YourName2_YourName2_Iteration2.zip**. The two files include:
   a. Hi-rez copy of your updated class diagram (PNG or JPG). I must be able to zoom in and read it comfortably. Name it YourName_Iteration2_Class.PNG (or YourName1 etc if team of 2).
   b. Your writeup as a Word DOC or PDF. Name it YourName_Iteration2_Writeup.DOC (or YourName1 etc if team of 2).

      This should be the easiest bit of Iteration 2. Please make my life easier by doing this bit carefully. You get no points for doing it right, but you will lose at least 25% for doing it wrong.

      If you are a Team of 2, each of you should submit the **same** zip file. Note your team partner when submitting. This makes grading easier.

4. **TLDR**
   Be neat, professional, and thorough. Know the domain very well. Ask questions if you need information. If you think there is a mistake somewhere, ask. Do not wait until the last minute to do this.