CSI/CEN 213: [H1] Object Oriented Design

Sun, Feb 5
Sun, Feb 12

1 Overview

Over the next two weeks you will be designing and creating classes to represent things you would find in an old-time, text-based Role Playing Game. This assignment will be split into two parts: a planning component due Sun, Feb 5, and an implementation component due Sun, Feb 12.

2 Planning

Sun, Feb 5

Look over the specs provided and decide how to most effectively use object oriented design principles to create the required classes. Be sure to consider issues such as inheritance and method overloading in your design.

1. Create a Project Management Plan for completing your program before Sun, Feb 12.
   Your plan should *minimally* include a short-term deadline for
   - Creating a UML diagram for the program
   - Programming and testing each of the classes you'll need to implement.
   - Making a screen-cast to show that your program works.

   Don't try to schedule the entire programming portion of the project during the week of Sun, Feb 12. Instead, take a realistic look at your schedule: assignments for this class and other classes, extra curricular activities, employment, your need for sleep, and when office hours are scheduled. Give yourself deadlines that respect your entire schedule. At least some of them should be set for this week.

   `https://trello.com` and `https://asana.com` provide excellent free project management tools. They take different approaches, so pick the one that best fits your organizational style. Or try one this time and try the other one next time. If you have another project management tool that you prefer, you may use that instead. Take a screenshot of your project management plan then put it in a PDF to submit it to blackboard.

2. Create a UML diagram of your object oriented design.
   `https://www.draw.io` provides an excellent free resource for creating UML diagrams. If you have a tool for creating UML diagrams that you prefer, you may use that instead. Be sure to print/export/save your UML diagram as a PDF to submit it to blackboard.

3 Specification

Here is the specification for the public functions—assume that these need to work as specified to integrate with a larger project. You may choose to add any private methods that you think you will need.

Item	
getName()	Returns the String *name*.
getWeight()	Returns the integer *weight*.
examine()	Prints a description of the object, including *name* and *weight*.

Weapon	
getDamage()	Returns the integer *damage*.
getName()	Returns the String *name*.
getWeight()	Returns the integer *weight*.
examine()	Prints a description of the object, including *name* and *weight*.

Armor	
getDefense()	Returns the integer *defense*.
getName()	Returns the String *name*.
getWeight()	Returns the integer *weight*.
examine()	Prints a description of the object, including *name* and *weight*.

Food	
getNutrition()	Returns the integer *nutrition*.
getQuantity()	Returns the integer *quantity*.
setQuatity(int newQuantity)	stores *newQuantity* in *quantity*.
getWeight()	Returns the integer *weight*, taking the quantity into account.
getName()	Returns the String *name*.
examine()	Prints a description of the object, including *name* and *weight*.

4 Implementation

Sun, Feb 12

Write the program that meets the specifications provided. Use git for responsible version control. You should be able to rely on your UML diagram to decide how to approach your program. If you decide along the way that your UML diagram wasn't the best way to organize the program, it's OK to deviate from it.

In addition to creating the classes specified, you'll need to create a main method that thoroughly tests each function you build to show that it works. You should do these things in parallel, not save testing for the end. Here's a sense of how your workflow ought to go:

1. Write a function, documenting it as you go.
2. Write as many tests as you need in the main method to show that it works. Test both typical uses and atypical uses that might break it.
3. Debug as necessary.
4. Commit.

Some people find it easier to swap steps 1 and 2. You can write your tests first then go back and write your code later.

When you are done with your program, take screenshots of your git commits to demonstrate responsible version control. (See H0: Hello World for more information).

5 Screencast

In addition to submitting your java code, you will also submit a screencast in which you show your code and demo it. This lets you show us that your code works on your system. If your code does not compile, you will receive a 0. You do not need to talk for your screencast; you are just using it to demonstrate your code.

Use jing to create your screencast. Download jing here: `http://www.techsmith.com/jing.html`

In your screencast, you should do the following:

- slowly scroll through the file (don't talk over this part; just scroll slowly enough for someone to read)
- compile your program to demonstrate that it compiles
- run your program to demonstrate that it works.
- If necessary, expand your test window if there is a lot of output. If there is more output than fits on screen, scroll slowly through this output as well.

There is a demo screencast you can use as a model here: `http://screencast.com/t/KKnT8ibrUkkZ`.

When you are done recording your screencast, choose the option to share it at `http://screencast.com`. This will put the url in your clipboard so you can paste it into your blackboard submission. Do not try to download the screencast and submit that file; just submit the url.

Submit

Sun, Feb 5

- ☐ Attach your UML design in PDF format.
- ☐ Attach screenshots of your Project Management Plan in PDF format.

Sun, Feb 12

- ☐ In the write submission field, provide a link to your screencast at http://screencast.com.
- ☐ Attach your .java files. If there is just one, you don't need to compress it. If there are multiple files as part of the assignment, compress the project homework for the assignment using .zip. Do not use .rar or some other form of compression.
- ☐ Attach the PDF containing your screenshots.