CS 249: Assignment 5

Thinking in Objects

Theory Questions (14%)

Note: for any UML diagrams, you can use a UML drawing utility like UMLet.

1. (2%) Draw the UML diagram for the class defined in Programming Exercise 10.11 at the end of Chapter 10. **Enforce encapsulaton!**

2. (2%) Design the following two classes for a top-down 2D game (similar to the NetHack example from before).

   You will draw UML diagrams for both of these classes.

   First, design a class for a **Weapon**.
   It should contain the following data:

   - Name
   - Damage

   It should have the following functionality:

   - Constructor that takes name and damage
   - Getter/setter functions for data
   - STATIC function that takes a Weapon and prints the name (if null, print ”No weapon”)

   Next, design a class for a **Player**.
   It should contain the following data:

   - Position in X and Y (you may assume integer coordinates)
   - Health (defaults to 100)
   - Current Weapon (defaults to null)

   It should have the following functionality:

   - Constructor that takes position
   - Getter/setter for position
   - Getter/setter for health
   - Getter/setter for current Weapon

   Ensure that your classes are:

   - Clear, consistent, and have central purpose
   - Maintaining a good abstraction
   - Enforcing encapsulation

3. (2%) Given the Player and Weapon classes (also look at Programming Requirement 2), what do you think most accurately represents the relationship between Player and Weapon?

   (a) Association

   (b) Aggregation

   (c) Composition

4. (2%) Given your answer, draw the UML diagram that shows the relationship between these classes. **Use a simplified form of the UML class diagram that just includes the name.** Also, **be sure to include the correct numbers on either end of the line!**

5. (2%) Which of the following is **NOT** part of a **class contract**?

   (a) Description of how public parts are expected to behave

   (b) Private methods and fields

   (c) Public constructors, methods, and fields

6. (2%) An **Abstract Data Type (ADT)** is 1) a collection of data and 2) operations that work on that data.

   (a) True

   (b) False

7. (2%) Given the code below, what is this an example of?

   (a) Autoboxing

   (b) Autounboxing

   Integer x = 5;

Programming Assignments (86%)

Ensure you are enforcing encapsulation!!!

For this assignment, use a SEPARATE Java file for THE TEST PROGRAM of each requirement (not sub-requirements)! Name these Java files "Assign5_N.java", where N is the requirement number.

Note that you will also create other classes; these should be named "ClassName.java", where ClassName is the name of the public class inside the .java file.

#	*Questions*	
1	**Implement the code necessary for Programming Exercise 10.11 at the end of Chapter 10.**	
	Hint: Consider making a private function to get the distance from a given point to the center of the circle.	
	OUTPUT: Area = 95.03317777109123 Perimeter = 34.55751918948772 Contains (3,3)? true Contains Circle2D(4,5,10.5)? false Overlaps Circle2D(3,5,2.3)? true	
2	**Implement the two classes you defined in Theory Question 2, but also implement a class Assign5_2 that tests these classes. Make the best use of the functions that are available to you in the Weapon and Player classes. In its main method:**	
	Create a Weapon instance, sword, with name "Common Sword" and damage "50".	
	Create a Weapon instance, coolSword, with name "Glamdring, the Foe-Hammer" and damage "1500".	
	Create a Player instance, player, with position (100,100).	
	Print the current weapon of the player. (Should output "No weapon")	
	Set the player Weapon to be "sword".	
	Print the current weapon of the player. (Should output "Common Sword")	
	Set the player Weapon to be "coolSword".	
	Print the current weapon of the player. (Should output "Glamdring, the Foe-Hammer")	

Submission

You will submit the following items as a *.tar or *.zip file:

- A plaintext, Word doc, or PDF with your answers to any theory questions
- Your .java file(s)

Submit this tar/zip file on Blackboard under the appropriate assignment.

Do NOT submit:

- Your .class file(s)
- Your project files

Grading

Below is a list of SOME of the grading penalties:

- Submitting ONLY .class files and NOT .java files
- Sloppy or poor coding style
- Bad coding design principles
- Code that does not compile
- Code that crashes, does not run, or takes a VERY long time to complete
- Using code from ANY source other than the course materials
- Collaboration on code of ANY kind; this is an INDIVIDUAL PROJECT
- Sharing code with other people in this class or using code from this or any other related class
- Output that is incorrect
- Output that is NOT generated by the proper algorithms
- Algorithms/implementations that are incorrect
- Submitting improper files
- Failing to submit ALL required files