Practices for Lesson 2

Practices Overview

In these practices, you first analyze a problem using object-oriented analysis, and then you design a possible solution by using UML-like notation.

Practice 2-1: Analyzing a Problem Using Object-oriented Analysis

Overview

In this practice, you analyze a case study and use object-oriented analysis to list the objects, attributes, and operations in the case study.

Preparation

Read the following case study, and then model the system by choosing objects and their attributes and operations.

Case Study

A soccer league needs a system to track team and player standings.

At any moment, administrators want to be able to report a list of games played with results, a list of teams ranked by wins, and a list of players on each team ranked by goals scored.

Tasks

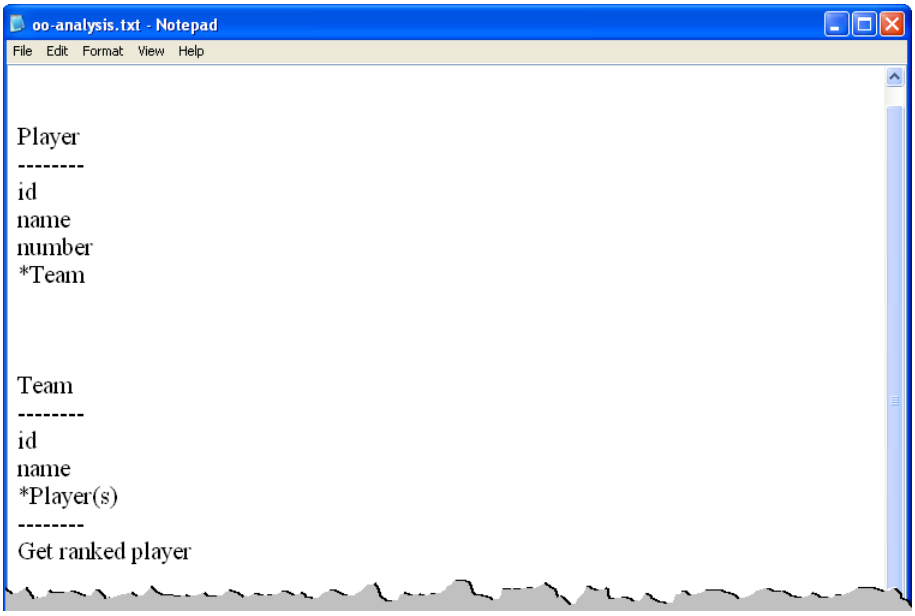
Your task is to produce an object-oriented analysis for a Java technology application that tracks soccer scores. The program should track:

- The list of *players* on each *team* ranked by *goals* scored
- The list of *games* played with results
- The list of teams in the *league* ranked by wins

Hint: You can think of the objects as nouns, attributes as adjectives, and operations as verbs. As an example, a Player is a noun, the player's name is an adjective that describes that noun, and add goal is a verb.

The application should be able to generate statistics for teams, players, and seasons.

1. Open the text editor by selecting Start > Programs > Accessories > Notepad.
2. Save the file
3. To get started, list the high-level classes that are included in this problem. You can list them in the text editor and use dashed lines to separate the objects, attributes, and operations as shown in the screenshot.



4. (Optional) You can use the UMLet tool if you choose. Double-click the UMLet icon 

from the Windows desktop to launch the program.

Solution

Player	Team	Game	League	Goal
id name number *Team	id name *Player(s)	id team one score team two score *Goal	*Team(s) *Game(s)	id *Team *Player time
	Get ranked player	Get results	Get game results Get ranked teams	

The asterisk (*) denotes attributes that are also objects.

Practice 3-2: Designing a Programming Solution

Overview

In this practice, you continue with Practice 3-1 by using UML-like notation to represent the classes you identified.

Assumptions

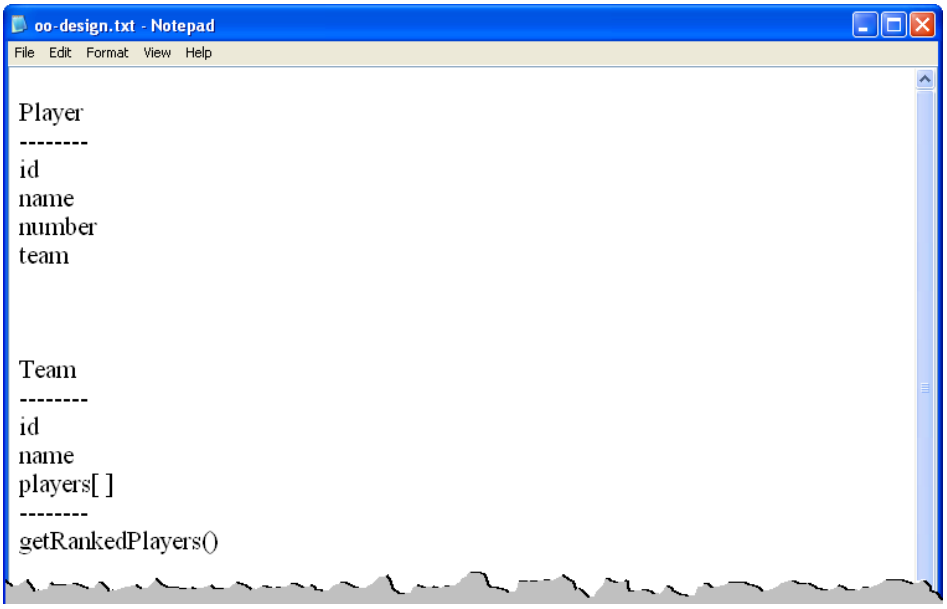
You have completed identifying the objects, attributes, and operations that you found in Practice 3-1.

Tasks

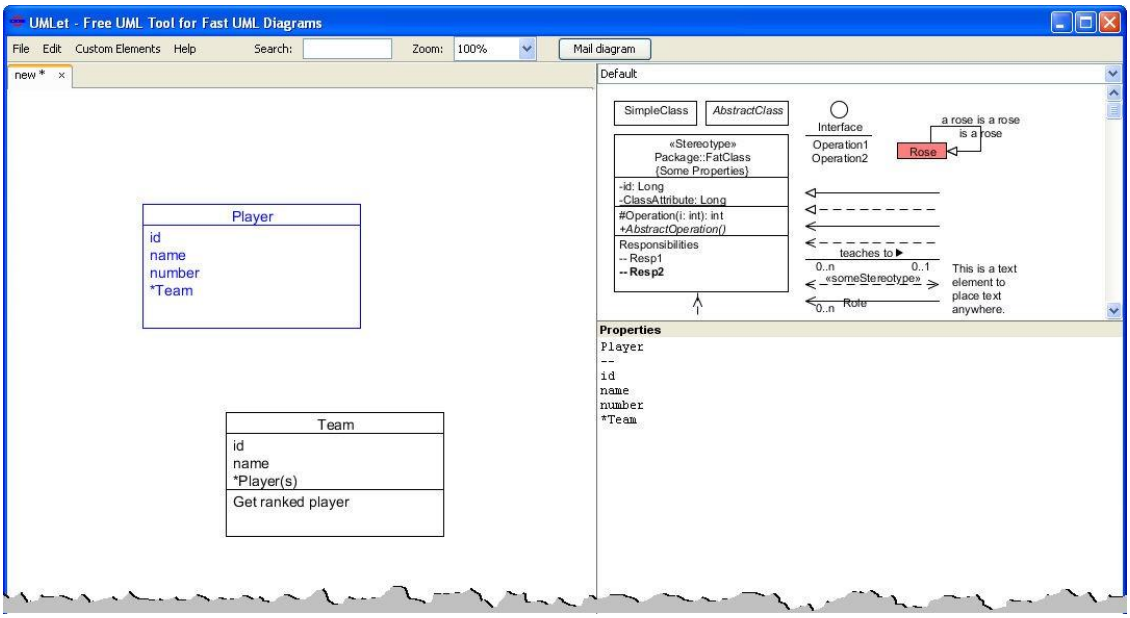
Your task is to produce a design for each of the classes in the earlier system for tracking soccer scores. Remember to:

- Use camelCase to name your classes, attribute variables, and methods
- Identify a valid range of values for each attribute (where a range is known)
- Use square brackets to indicate an attribute that represents a collection of values (players[ ])
- Use parentheses to identify methods

1. Open your file from previous practice
2. Use the classes, variables, and operations that you identified in the previous practice, and develop method names for the operations. The screenshot below is an example.



3. (Optional) You can use the UMLet tool if you choose. Double-click the UMLet icon from the Windows desktop to launch the program.



Solution

Player	Team	Game	League	Goal
id name number team	id name players[ ]	id team one score team two score goals[ ]	teams[ ] games[ ]	id team player time
	getRankedPlayers()	getResults()	getGameResults() getRankedTeams()	

Note: Although not shown in the solution, you need add/remove methods for each collection attribute and get/set methods for all other attributes. We have not discussed those methods at this point in the course.

Your solutions might look different from the suggested solution. The purpose of this lesson is to help you continue thinking in terms of objects, attributes, and operations. You have another opportunity during this course to practice modeling a programming solution.