NAU NORTHERN ARIZONA UNIVERSITY
School of Informatics, Computing, and Cyber Systems

CS 440 Software Architecture
Group Assignment A-1.4 - In-class
(Mental models)

Groups: 1 to 3 students (does not need to be the same as the group project).
Delivery: Each student should individually turn in the solution as a **PDF** on Canvas.
Instructions: Make a copy of this document and share it with group members.

Goal: Discuss code documentation.

0) Who worked on this solution:

Name
Christian Lamb
McKay Hartman
Jackson Belzer

Discuss the following questions within your group. Write a short answer. Be prepared to share your insights with the rest of the class.

Everyone in the group should replicate the activity on their machines.

Tracking: Mark that you started it in this shared spreadsheet so we can keep track of how the groups are progressing.

A-1.4 Activiy_Tracker

Representing a feature

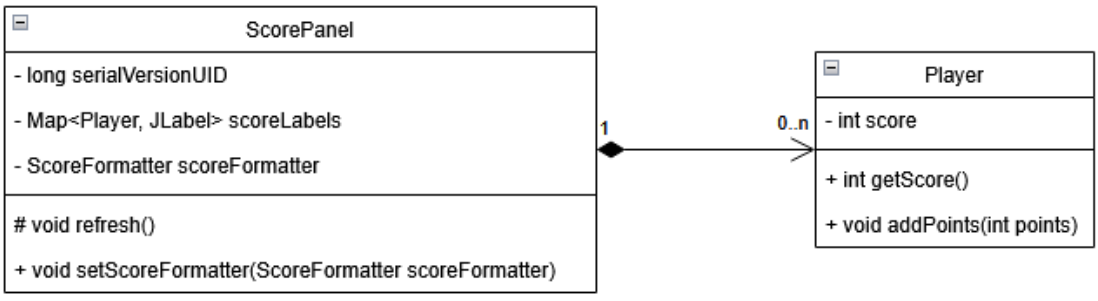
1. **You should still have a clone of JPacMan3. If not, clone the code from https://github.com/SWE-265P/jpacman3**

   Find all the places where the code deals with scoring. Draw a model that represents this part of the code. For now, we will not prescribe a specific format for this model. We will let you experiment in

teams—there is no right answer as to what your picture will look like.

The goal of such a model is to help someone understand where this feature is implemented so that they can perform a refactoring or use that feature in a different system. Think of the model as a map.

1.1. Add your model to the box below.



1.3 What representation did you choose for your model? Why?

The representation we chose for the model was a standard UML diagram format that shows where the score is displayed to the user and the place where the score is kept and managed, which is in the Player, specifically highlighting the relevant parts of the Player class that deals with scoring. We also show that a score panel can have many players but is reliant on the Player or Players existing.

1.4 What strategies did you use to understand the code?

We used a bottom up approach, starting at the specific variables and functions that we found relating to scoring and then expanding our search outward to other relevant parts of the code to see how they connect.

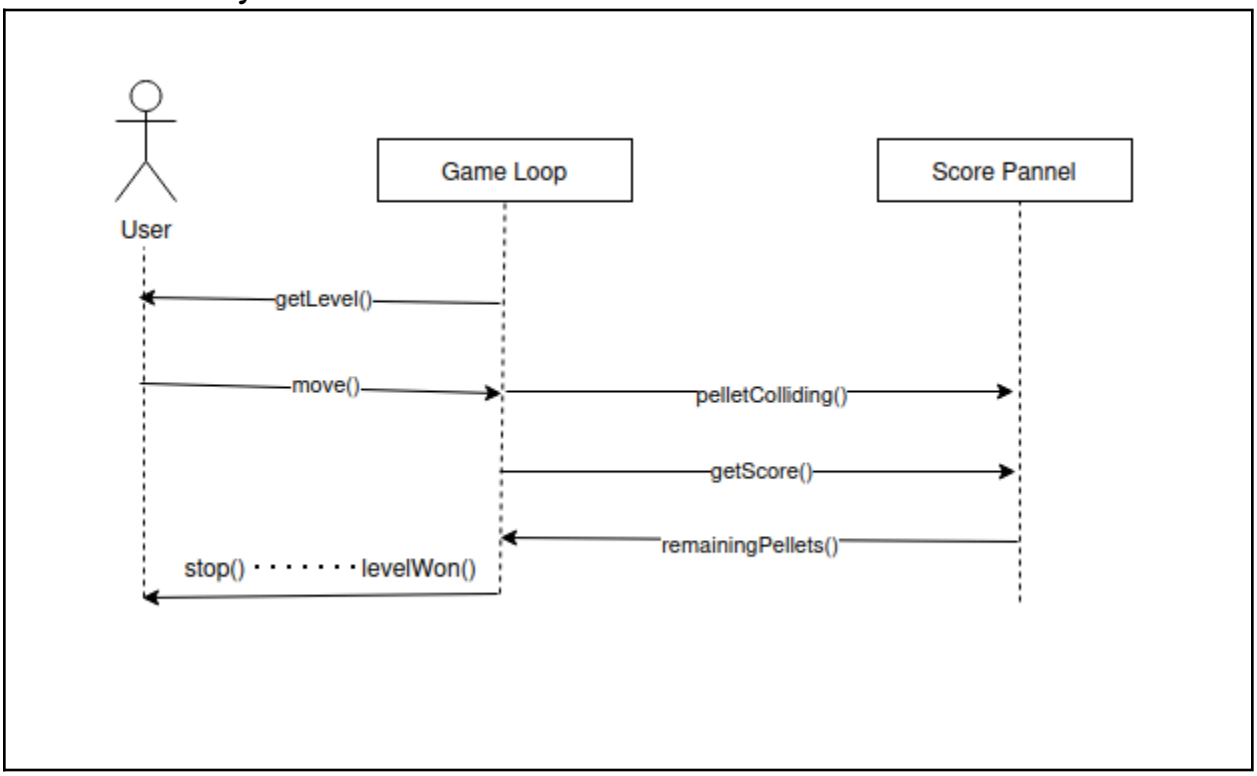
Tracking: Mark that you completed Part I in this shared spreadsheet so we can keep track of how the groups are progressing.

A-1.4 Activiy_Tracker

2. **Draw a model of how scoring works. This model is different than the previous one, which was more focused on the structure (a map of where the feature was located). This model should represent the**

dynamism of the scoring. Someone will be able to understand how the different parts of the code interact to produce the score.

2.1. Add your model to the box below.



2.2. What representation did you choose for your model? Why?

For this model we chose a UML sequence diagram. Sequence diagrams show structure and interaction between different modules of code. Sequence diagrams are more oriented towards methods and show how the code works when the program is running.

2.3. Are both models necessary? Why?

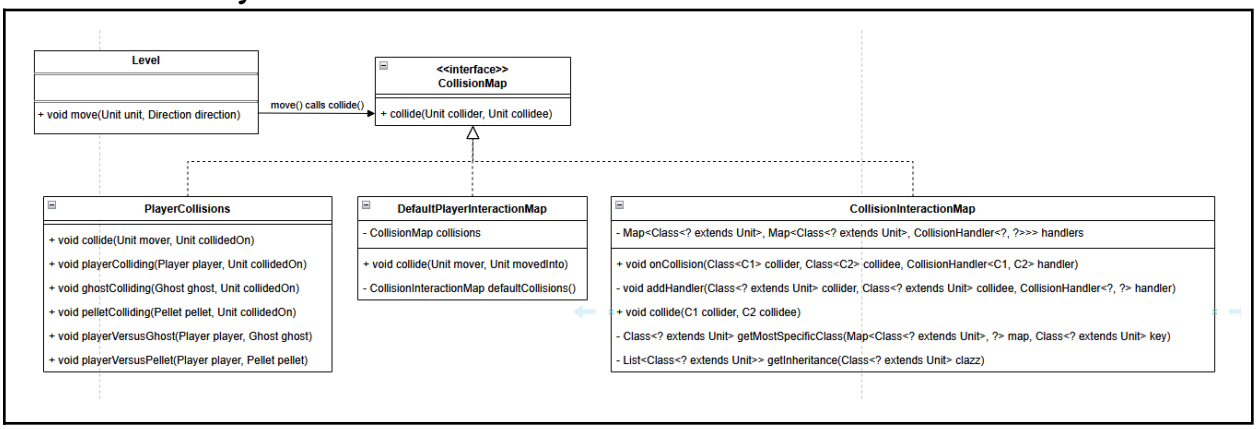
Yes, both models are necessary because class diagrams are more for displaying the individual classes and how they interact, while sequence diagrams show the interactions between actors and the program.

Tracking: Mark that you completed Part II in this shared spreadsheet so we can keep track of how the groups are progressing.

A-1.4 Activiy_Tracker

3. **Draw a model of how collisions work. You can choose to represent the structure of the code, the dynamics of the interaction among components, or both.**

   3.1. Add your model to the box below.



Tracking: Mark that you completed Part III in this shared spreadsheet so we can keep track of how the groups are progressing.

A-1.4 Activiy_Tracker