TP: MONOPOLY-LIKE MECHANICS

PABLO TESONE

1. Introduction

This TP aims to implement a non-trivial problem in Object-Oriented Programming. You will have to implement in Pharo or Java all the requirements included in this document.

2. Problem Context

The problem we require you to model is a simplification of the mechanics of the Monopoly game[1]. We are centring the problem in the buying, renting, and mortgaging. The problem includes the handling of railroads, terrains, houses, hotels and the monopolies you can build with the combination of them.

2.1. **General Rules.** The rules for the game can be found in the rules page of the Monopoly Fandom Wiki[2].

From this set of rules we are going to implement the following use cases:

- Buying a property (terrain, railroad or utility).
- Asking a mortgage over a property / Paying back the mortgage.
- Buying houses and hotels.
- Calculating the rent amount for a given property (for terrains, railroads, and utilities). It should consider the monopoly situations, the houses, hotels and mortgages.

3. Test Cases

You should implement the test cases that you consider that covers the main logic of each of the previous use cases. As a friendly reminder, each test should have a clear objective (maybe dividing tests is a good idea) and asserts (a test without asserts is not a test).

4. Administrative Considerations

The TP should be sent by the 31/10/2024. Any later sending will be not taken into account. The TP should be sent as a single zip file with the content of the project.

For Pharo solutions, you can export the whole package doing right-clicking and selecting 'File Out' generating a single file with all the code of the package, that is the file to put in the Zip.

For Java solutions, it is important to Zip the entire project.

[1] `https://en.wikipedia.org/wiki/Monopoly_(game)`

[2] `https://monopoly.fandom.com/wiki/Monopoly_Rules`

You might include any other document you consider important to explain your solution, *e.g.* if you have taken any special consideration or understanding of a given point.

In all the cases, the delivered code should compile and work. Non-compiling code will be automatically rejected.

5. About Doubts and Different Interpretations

All the rules, descriptions and requirements are open to different understanding. If a point is unclear or it raises different alternatives, you are free to choose the one that better suits you. The only condition is to explain in the affected part of the TP that you have taken a given direction and why you have done it.