Group 109: Jonathan V, Stephnie R, Brady H, Eyadame

Team name: Terp Coders

Aric Bills

INST326

12/11/2025

INST326: UMD Monopoly Final Submission

1. **Purpose** of each file:

- UMD_player.py
  - This file models each of the players of the monopoly game. It maintains a player's cash, all their property owned, movement around the board, and all actions taken by the players. This includes buying properties and paying rent. The file also contains player-specific information. For example, each player's jail status, bankruptcy, and property tracking. Additionally, the file allows the user to save the game (JSON format) and later be able to load the game and play the saved version
- UMD_property.py
  - This file defines every type of property, whether it be campus or special category, on the UMD-themed Monopoly digital board game. Every property has its own price, rental amount, and an assigned group. The file implements the rules concerning how the rent is calculated, mortgaged, and invested in a property. All while engaging with the board file and managing the ties between the many different properties.
- board.py

- ➢ This file provides a text-based visual representation of the Monopoly game board. Creates an 11x11 grid layout that shows the properties of the campus and their positions, and shows the player tokens in their current location on the board. The specific board cells are made to be in liner game position, which makes it possible to track the players on the visual board.

- ❖ Decision_engine.py
  - ➢ The decision engine serves as an AI-based purchase recommendation system within the UMD Monopoly game. The engine assesses the appropriateness of property purchases by evaluating how many properties a player can r
- ❖ Event_generator.py
  - ➢ The file is capable of generating numerous random events for the game through the real-life occurrence at the University of Maryland which reflects the way life operates on campus. The players will experience both positive and negative occurrences in an alternating fashion (for example: "Did you really get a new scooter or car?" positive, or "I hope you didn't get a parking ticket" negative). The randomness supplied by the Event Generator provides players with a more realistic university experience by adding a level of uniqueness to the gameplay and subsequently can enhance the positive, and negative dynamics within the game.
- ❖ game.py

- ➢ This file is the principal controller of all aspects of the Monopoly game. This includes the players, properties, the game board, CPU decision making as well as the game events. It provides the player with a complete UMD Monopoly gaming experience. Contains information about the basic game cycle, order of turns, player interactions with the game board, as well as all aspects of the game play. Like, buying and renting properties and winning or losing the game. It presents the players with the choice to compete against the CPU or another player

❖ save.py

- ➢ Last but definitely not least. The save file allows users to be able to save their progress on the game board to a JSON file with timestamps. The save file serializes all the payer data(properties owned/rented and game stats) to disk. This makes it possible for players to then open the saved version at a later time to finish the game.

2. Clear instructions on **how to run your program** from the command line
    a. If your program takes command-line arguments, please document the command-line interface (which arguments are required? which are optional, if any? what data types are you looking for? are there a specific format for data files? etc.)
        i. From the project directory, in the terminal type “python3 game.py” in the terminal if on Mac or “python game.py” if on Windows.
3. Clear instructions on **how to use your program and/or interpret the output of the program**. Anything the user might encounter while using your program that a random

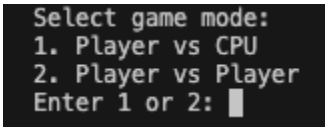
person on the street would not find self-explanatory needs to be explained in your instructions.

a. Make sure all project files are in the same directory.

b. Open a terminal in the folder.

c. To run the game, type “python3 game.py” in the terminal if on Mac or “python game.py” if on Windows.

d. Then you will be prompted to select a mode:

i. 

e. Enter player name(s). If you leave the name(s) blank, the program will automatically assign Player1 and Player2

f. Then the game will run and display:

i. Dice rolls

ii. Board updates

iii. Where players land

iv. Rent payments

v. Buy or skip prompt

vi. CPU decisions

vii. Event/special tiles

g. Game ends when someone is bankrupt or max turn limit has been reached. The program will then save the results of the game to a JSON file

4. An **annotated bibliography of all sources** you used to develop your project, including sources of data, sources of background information about your project topic, and sources about Python programming that informed specific aspects of your code. For each source, explain how you used the source. You do not need to cite any INST 326 course materials.

Sources:

1. https://maps.umd.edu/map/ : The map of the University of Maryland, College Park campus was used to double-check all the listed properties used in the game, and their locations. \
2. https://www.hasbro.com/common/instruct/monopoly.pdf : instructions on how to play monopoly

5. **Attribution**: in order to evaluate whether each member has made a substantial, original contribution to the project, please provide a table like the one below, with a separate row for each method or function.

Method/function	Primary author	Techniques demonstrated	File name
UMDProperty.find_best_investment	Stephnie Rozario	Key function with sorted()	UMD_property.py
Player.buy_property() Player.can_afford()	Stephnie Rozario	Conditional Expressions	UMD_player.py
__str__	Eyadama Gnassingbe	List comprehension, Magic Methods	game.py
event_generator	Eyadama Gnassingbe		event_generator.py
__init__	Jonathan Valladares	Optional parameters	game.py
take_turn	Jonathan Valladares	composition	game.py

decision_engine()	Brady Hillbert	f”” strings	decision_engine.py
display_board()	Brady Hillbert	Sequence Unpacking	board.py