PlayerThread waits for user input, executes a turn, and triggers events/encounters.

AI threads (The other characters) run autonomously, gathering loot, fighting enemies, and progress in the world.

The InputHandler reads console commands and sends the responses to the player thread

The Main thread creates characters, starts threads, and waits for them to finish.

For potential race conditions we have an issue where multiple characters may acquire loot simultaneously, along with a potential issue where the multiple threads requesting said loot could cause RNG order to cause unpredictable distribution. For fixes we have take() synchronized to ensure atomic access.

We have it where score and HP updates during combat. The current design keeps combat inside the local context of each character's fight. Since each battle is isolated, no conflicting writes occur unless two characters are fighting the same boss simultaneously, which shouldn't happen as each character has their own unique final boss, which brings us to Final Boss Triggers.

Multiple characters could reach the score threshold to trigger the final boss simultaneously which could cause two bosses to spawn and/or two endings to trigger, which I attempted to prevent from happening with separate story endings and bosses for each character.

For thread coordination we have Wait() which pauses the player until they issue a command. Notify() which resumes the player after mentioned input. Thread.sleep() which paces the events and actions for the AI characters, and Synchronized which is supposed to ensure thread safety during shared access.