Text Adventure Game Classes Documentation

Brief description

The structure of a text adventure game with three main classes: Scene, Event, and Game, as well as a Character class.

Game Class: This class is central to the game's logic. It contains attributes such as the current scene (currentScene) and the protagonist (protagonist). The Game class also defines methods to start (start()), load (load()), run (run()), and exit (exit()) the game.

Event Class: This class is responsible for triggering events within the game. It includes a method to trigger an event (trigger()) and another to return a string representation of the event (toString()).

Scene Class: The Scene class manages the different scenes in the game. It holds a list of events (List<Event>) and a map of characters (Map<String, Character>). It also includes methods to act within a scene (act()) and to convert the scene to a string representation (toString()).

Character Class: Although not the main focus, the Character class still plays a significant role in the game. It includes attributes like health, weapon, armor, and a description. The class also has various getters and setters.

Character Class

The base class for all characters in the game, encompassing the fundamental attributes and actions that every character possesses.

Attributes:

- 'health': Integer - The character's health points, determining how much damage they can sustain.
- 'weapon': Weapon - The weapon that the character is equipped with, influencing their attack capabilities.
- 'armor': Armor - The armor that the character is wearing, serving to reduce incoming damage.
- 'description': String - A textual description of the character, potentially including appearance and backstory.

Methods:

- 'getters and setters()': Varies - Methods to retrieve and set the above properties.

Protagonist Class (extends Character)

Represents the main player character in the game, with additional properties and actions.

Attributes:

- 'magic': Magic - The magical abilities possessed by the protagonist, used for casting spells.
- 'money': Integer - The amount of currency the protagonist has, utilized for transactions.
- 'skills': List<Magic> - A list of magical skills the protagonist has mastered.
- 'possessions': List<Goods> - A list of items in the protagonist's possession.
- 'experience value': Integer - The experience points of the protagonist, used for leveling up and skill acquisition.

Methods:

- 'getters and setters()': Varies - Methods to retrieve and set the protagonist-specific properties.

Monster Class (extends Character)

Represents monsters or antagonistic creatures in the game, offering challenges and rewards to the protagonist.

Attributes:

- 'gold': Integer - The amount of gold carried by the monster, potentially claimable by the protagonist upon defeat.
- 'goods': List<Goods> - A list of items that the monster may drop when defeated.

Methods:

- 'getters and setters()': Varies - Methods to retrieve and set monster-specific properties.

Magic Class

Defines the properties and mechanics of magic within the game, including spells and abilities.

Attributes:

- 'description': String - Describes the magic's effects and lore.
- 'value': Integer - Represents the strength or impact of the magic.
- 'cost': Integer - Indicates the resource cost to utilize the magic.

Methods:

- 'getters and setters()': Varies - Methods to retrieve and set magic properties.

Game Class

The central class of the game that manages the game state, including the current scene and the protagonist.

Attributes:

- 'currentScene': Scene - The scene currently active in the game context.
- 'protagonist': Protagonist - The main character that the player controls.

Methods:

- 'start()': void - Initializes the game environment and starts the gameplay.
- 'load()': void - Loads a saved game state.
- 'run()': void - Runs the game loop, allowing for continuous gameplay.
- 'exit()': void - Closes the game and exits the application.

Event Class

A generic class for events in the game, providing a framework for event triggering and description.

Methods:

- 'trigger()': void - Triggers the implementation-specific event action.
- 'toString()': String - Returns a string representation of the event for debugging or display purposes.

AttackEvent Class (extends Event)

Handles the logic for attack events in the game where one character engages in combat with another.

Attributes:

- 'attacker': Character - The character initiating the attack.
- 'attackee': Character - The character being attacked.

Methods:

- 'triggerFight()': void - Specific method to execute the combat mechanics between attacker and attackee.
- 'trigger()': void - General method to trigger the attack event.

TradeEvent Class (extends Event)

Encapsulates the trade interaction between two characters, facilitating the exchange of goods.

Attributes:

- 'buyer': Character - The character attempting to acquire goods.
- 'seller': Character - The character offering goods for sale.
- 'goods': Goods - The item or items being traded.

Methods:

- 'triggerTrade()': void - Method to execute the trade between the buyer and seller.
- 'trigger()': void - General method to trigger the trade event.

MagicEvent Class (extends Event)

Represents an event where a character uses a magical ability or spell.

Attributes:

- 'caster': Character - The character using the magic.
- 'target': Character - The character or entity targeted by the magic.
- 'magic': Magic - The magic spell or ability being used.

Methods:

- 'triggerMagic()': void - Method to execute the magical effect on the target.
- 'trigger()': void - General method to trigger the magic event.

TaskEvent Class (extends Event)

Describes an event where a character undertakes a task or quest.

Attributes:

- 'character': Character - The character associated with the task.
- 'task': Task - The task or quest to be undertaken.

Methods:

- 'triggerTask()': void - Method to initiate the task for the character.
- 'trigger()': void - General method to trigger the task event.

MoveEvent Class (extends Event)

Defines an event that handles the movement of characters between different scenes in the game.

Attributes:

- 'character': Character - The character that is moving.
- 'moveTo': Scene - The new scene to which the character is moving.

Methods:

- 'triggerMove()': void - Method to execute the character's movement to a new scene.
- 'trigger()': void - General method to trigger the move event.

Scene Class

Defines a generic scene in the game, which can contain characters, items, and events.

Attributes:

- 'events': List<Event> - A collection of events that can occur within the scene.
- 'characters': Map<String, Character> - A map of characters present within the scene, keyed by a unique identifier.

Methods:

- 'act()': void - Represents an action taken by a character within the scene.
- 'toString()': String - Provides a textual description of the scene.

TownScene Class (extends Scene)

A specific type of scene representing a town where characters can interact with task boards and marketplaces.

Attributes:

- 'taskBoard': List<Task> - A list of tasks that characters can accept while in the town.
- 'marketplace': List<Goods> - A collection of goods available for trade in the town's marketplace.

Methods:

- 'act()': void - Overridden method for actions specific to the town scene.
- 'actTask()': void - Method to engage with tasks on the task board.
- 'actTrade()': void - Method to engage in trade at the marketplace.

FieldScene Class (extends Scene)

Represents an outdoor scene in the game, typically involving exploration and encounters with items or enemies.

Attributes:

- 'fieldItem': Map<String, Goods> - A mapping of items that can be found in the field.

Methods:

- 'act()': void - Overridden method for actions specific to the field scene.
- 'actFight()': void - Method to initiate combat in the field.
- 'actPickUp()': void - Method to collect items found in the field.

Goods Class

The base class for all items in the game, which can be traded, used, or collected.

Attributes:

- 'description': String - The textual description of the goods, detailing its appearance or use.
- 'amount': Integer - The quantity of the item available or required.

Methods:

- 'toString()': String - Returns a string description of the goods for display purposes.

Weapon Class (extends Goods)

Represents weapons in the game, used by characters to increase their attack power.

Attributes:

- 'attackPower': Integer - The measure of how powerful the weapon is in combat.

Armor Class (extends Goods)

Represents armor in the game, used by characters to protect themselves and reduce incoming damage.

Attributes:

- 'defensivePower': Integer - The measure of the armor's ability to absorb or deflect attacks.