“Labyrinth game”- documentation

Created by Kacper Toczek

About project:

The program I wrote is a game. The player is placed in a labyrinth and has to find the exit. During the game, the player collects valuable items and keys which he can use to open next chambers. In chambers there are hidden monsters with which the player fights if he enters the chamber.

Classes included in the program are:

- Labyrinth
- Player
- Chamber
- Door
- Item
- Key
- Duel
- Monster
- Task

The program also includes a loop that simulates the game. It allows you to play without programming knowledge.

Description of individual classes:

1) **Labyrinth**
   This is the main class. It is responsible for creating the labyrinth and adding to it the necessary components such as chambers or player.
   a) Attributes:
      i) **name** – used to describe the labyrinth
      ii) **player** – the labyrinth remembers the player in it
      iii) **player_name** – used to create a player
      iv) **chambers** – it is a list of chambers, the player can move there
      v) **tasks** – it is a list of tasks from which the monster chooses the task for the player during the duel
      vi) **map** – it is a map of labyrinth
   b) Methods:
      i) **get_description()** – returns the labyrinth description
      ii) **add_player()** – adds a player to the labyrinth who will be there until the end of the game
      iii) **add_chambers()** – adds a chamber to chambers list
      iv) **add_task()** - adds a task to tasks list
      v) **create_labyrinth()** – creates all chambers and components that are in them, can be modified by programmer to change the look of the labyrinth

2) **Player**

It is one of the most important classes in program. It allows the player to navigate through the labyrinth and do various activities.

a) Attributes:
   i) **name** - used to describe game
   ii) **level** – it is a level of player experience that changes during duels
   iii) **labyrinth** – the player remembers the labyrinth in which he is
   iv) **current_chamber** – it is a chamber where the player is currently located
   v) **items** – it is a list of items that the player collects during the game

b) Methods:
   i) **get_info()** – returns player description e.g. name, level
   ii) **get_items()** – returns items and their description that player has
   iii) **value_of_items()** – returns value of items the player has
   iv) **look_around()** – returns the description of chamber the player is in
   v) **open_door()** – if the player has a door key, it changes the status of the door from closed to open.
   vi) **check_doors()** – returns a description of the doors in the current chamber, which doors are closed, which doors are open
   vii) **take_item()** – transfers an item from current chamber to the player's backpack (list of items)
   viii) **drop_item()** - transfers an item from player's backpack (list of items) to the current chamber

3) **Chamber**

This class allows you to add a necessary components to the chambers e.g. doors, items or monster.

a) Attributes:
   i) **name** – used to describe the game
   ii) **start** – information where the player starts the game
   iii) **final** - information where the player ends the game
   iv) **labyrinth** – the chamber remembers the labyrinth to which it belongs
   v) **doors** – it is a list of doors the player can go through in this chamber
   vi) **items** – it is a list of items which the player can take from here
   vii) **monster** - it is a monster who will fight with the player
   viii) **duel** – it is a duel that will take place between player and monster

b) Methods:
   i) **add_door()** – adds a door to doors list
   ii) **add_item()** – adds an item to items list
   iii) **add_monster()** – adds a monster to chamber
   iv) **get_description()** – returns a chamber description
   v) **get_items()** – returns description of the items in chamber
   vi) **get_doors()** – returns description of the doors in chamber

4) **Door**

This class creates connection between objects of the Chamber class. The player uses them to change the chamber.

a) Attributes:
   i) **name** – used to describe the game
   ii) **needed key** – information which key is needed to open the door

iii) **connection** – information about connected chambers (from , to)
iv) **status** – closed/open information if the player can go through the door

b) Methods:
i) **get_needed key()** – returns the needed key to open door
ii) **check_status()** – returns the status of the door
iii) **change_status()** – if the player has a door key, it changes the status of the door from closed to open

5) **Item**
This class creates items which are placed in chambers.
a) Attributes:
i) **name** – used to describe the game
ii) **value** – used to describe the game, allow the player to compare the items
b) Methods:
i) **get_description()** – returns description of the item

6) **Key**
This class creates keys which are placed in chambers and allows the player to open doors.
a) Attributes:
i) **name** – used to describe the game
ii) **value** – used to describe the game, allows the player to compare the items
iii) **which_door** – information which door this key opens
b) Methods:
i) **get_with_doors()** - returns information which door this key opens
ii) **get_description()** - returns description of the key

7) **Duel**
The duel takes place when the player enters the chamber with the monster.
This class creates a duel and indicates the winner and the loser.
a) Attributes:
i) **player** – the duel remembers the player
ii) **monster** – the duel remembers the monster
iii) **task** – draws a task from task list
iv) **winner** – monster/player depends on player's answer
v) **loser** – monster/player depends on player's answer
b) Methods:
i) **ask_question()** – returns description of monster and question for the player
ii) **check_answer()** – checks the correctness of the answer and determines the winner and the loser
iii) **give_prize()** – change of the player's level depends on result of the duel

8) **Monster**
This class creates monsters which fight with the player.
a) Attributes:
i) **name** – used to describe the game
ii) **level** – used to change player's level
b) Methods:
i) **introduce_yourself()** – returns description of monster

9) **Task**
   This class allows you to create questions that the monster asks the player during the duel.
   a) Attributes:
      i) **task** – it is a task order
      ii) **answer** – it is a correct answer to the task
   b) Methods:
      i) **get_task()** – returns a task order
      ii) **get_answer()** – returns a correct answer