CIS*3260 A1 - Mazen Bahgat (1157821)

Class Description Submission

Credit:

Noureldeen Ahmed (1148182) provided all the use case and sequence diagrams. My code and class description submissions are solely based on my interpretation of his design.

Class Description: DropDead

Class Name:

DropDead

Single Responsibility:

API gateway to forward all requests to the DropDead model interfaces.

Instance Variables:

Variable Name	**Description**
dice_factory	DiceFactory object that configures the dice
point_counter	PointCounter object that counts the points a player scores.
score_keeper	ScoreKeeper object that keeps record of points scored by players.
coordinator	Coordinator object that handles players' action requests.

Method: initialize

- **Single Responsibility:** Initialize object instance state
- **Signature:** def initialize()
- **Arguments:** N/A
- **Description:** This method initializes all the instance variables listed above.

Method: configure_dice

- **Single Responsibility:** Route the request to @dice_factory object.
- **Signature:** def configure_dice(dice_count, sides)
- **Arguments:**
  - dice_count: Number of dice later created for each player's DiceSet
  - sides: Number of sides per dice created
- **Description:** Once the method is called, it routes the request to the *dice_factory* instance variable object by calling its *configure_dice* method using the same arguments.

Method: invite_player

- **Single Responsibility:** Route the request to @coordinator object.
- **Signature:** def invite_player()
- **Arguments:** N/A
- **Description:** Once the method is called, it routes the request to the *coordinator* instance variable object. It does that by calling coordinator's *add_player* method injecting dependencies using initialized instance variables.

Method: get_player_scores

- **Single Responsibility:** Route the request to @score_keeper object.
- **Signature:** def get_player_scores()
- **Arguments:** N/A
- **Description:** Once the method is called, it routes the request to the *score_keeper* instance variable object by calling its *get_player_scores* method.

Method: get_player_status

- **Single Responsibility:** Route the request to @coordinator object.
- **Signature:** def get_player_status(player_id)
- **Arguments:**
  - player_id: The player's ID in which the request is asking the status for.
- **Description:** Once the method is called, it routes the request to the *coordinator* instance variable object by calling its *get_player_status* method using the same arguments.

Method: perform_turn

- **Single Responsibility:** Route the request to @coordinator object.
- **Signature:** def perform_turn(player_id)
- **Arguments:**
  - player_id: The player's ID in which the request is performing the turn for.
- **Description:** Once the method is called, it routes the request to the *coordinator* instance variable object by calling its *perform_turn* method using the same arguments.

Class Description: Coordinator

Class Name:

Coordinator

Single Responsibility:

Coordinate player requested game state changes

Instance Variables:

Variable Name	Description
players	Hash of player IDs mapping to their respective PlayerModel instances.

Method: initialize

- **Single Responsibility:** Initialize object instance state
- **Signature:** def initialize()
- **Arguments:** N/A
- **Description:** This method initializes an empty players' hash set.

Method: add_player

- **Single Responsibility:** Add a new player to the game
- **Signature:** def add_player(score_keeper, point_counter, dice_factory)
- **Arguments:**
  - score_keeper: ScoreKeeper instance that keeps record of player's scores. Routed for use to the new PlayerModel instance.
  - point_counter: PointCounter instance that computes score points from a player's turn. Routed for use to the new PlayerModel instance.
- dice_factory: DiceFactory instance that initializes a new DiceSet for the new player. Routed for use to the new PlayerModel instance.
- **Description:** Once the method is called, initializes an :Active or :WAITING player status depending on current players state. This status along with the request arguments are used to initialize a new PlayerModel using its *create_player* method.

Method: get_player_status

- **Single Responsibility:** Retrieve the player's status from requested PlayerModel's given ID.
- **Signature:** def get_player_status(player_id)
- **Arguments:** N/A
- **Description:** Once the method is called, it looks for the target player's PlayerModel instance in the *players* hash set. The status is retrieved from the PlayerModel using its *get_player_status* method.

Method: get_player_scores

- **Single Responsibility:** Route the request to @score_keeper object.
- **Signature:** def get_player_scores()
- **Arguments:** N/A
- **Description:** Once the method is called, it routes the request to the *score_keeper* instance variable object by calling its *get_player_scores* method.

Method: get_player_status

- **Single Responsibility:** Retrieve a player's status from *PlayerModel* using its ID.
- **Signature:** def get_player_status(player_id)
- **Arguments:**
  - player_id: The player's ID in which the request is asking the status for.
- **Description:** Once the method is called, it routes the request to the *coordinator* instance variable object by calling its *get_player_status* method using the same arguments.

Method: perform_turn

- **Single Responsibility:** Perform a player's turn.
- **Signature:** def perform_turn(player_id)
- **Arguments:**
  - player_id: The player's ID in which the request is performing the turn for.
- **Description:** Once the method is called, it retrieves the player's PlayerModel instance using the provided ID. It routes the request to the *PlayerModel* instance variable object by calling its *perform_turn* method. Control is given to the next player to end the current turn.

Method: set_next_player_status

- **Single Responsibility:** Set the status of the next player to Active.
- **Signature:** def set_next_player_status(player_id)
- **Arguments:**
  - player_id: The current player's ID in which we're ending the turn for.
- **Description:** Once the method is called, it sets the next alive player's status to Active.

Class Description: PlayerModel

Class Name:

PlayerModel

Single Responsibility:

Handle a specific Player's state.

Instance Variables:

Variable Name	**Description**
id	Unique identifier for the player.
status	Label indicating the current status of the player. Active when its player's turn. Waiting when it's another player's turn and Dead when the player is DroppedDead.
dice_set	DiceSet object used by the player to perform their turns.
point_counter	PointCounter object used for counting the player's points scored in a turn.
score_keeper	ScoreKeeper object used to keep score after a player's turn.

Method: initialize

- **Single Responsibility:** Initialize object instance state
- **Signature:** def initialize()
- **Arguments:** N/A
- **Description:** This method initializes all instance variables with empty values.

Method: create_player

- **Single Responsibility:** Setup state to participate in a game.
- **Signature:** def create_player(status, point_counter, score_keeper, dice_factory)
- **Arguments:**
  - status: Initial player status
  - score_keeper: ScoreKeeper instance that keeps record of player's scores.
  - point_counter: PointCounter instance that computes score points from a player's turn.
  - dice_factory: DiceFactory instance that initializes a new DiceSet for the new player.
- **Description:** Once the method is called, it mutates its instance variables copying all the arguments.

Method: perform_turn

- **Single Responsibility:** Perform the turn
- **Signature:** def perform_turn()
- **Arguments:** N/A
- **Description:** Once the method is called, it rolls the associated dice set. The point counter counts the points scored, which is added to the player's score by the score keeper.

Method: drop_dead

- **Single Responsibility:** Kills the player
- **Signature:** def drop_dead()
- **Arguments:** N/A
- **Description:** Once the method is called, it mutates the *status* instance variable setting it to Dead.

Method: get_status

- **Single Responsibility:** Retrieve player status
- **Signature:** def get_status()
- **Arguments:** N/A
- **Description:** Once the method is called, it returns the *status* instance variable.

Method: id

- **Single Responsibility:** Retrieve player ID
- **Signature:** def id()
- **Arguments:** N/A
- **Description:** Once the method is called, it mutates the *id* instance variable setting it to Dead.

Method: set_status

- **Single Responsibility:** Mutates the status of the player.
- **Signature:** def set_status(status)
- **Arguments:**
  - status: The new player status
- **Description:** Once the method is called, it mutates the *status* instance variable setting it to the status argument.

Class Description: PointCounter

Class Name:

PointCounter

Single Responsibility:

Count player's scored points.

Instance Variables:

Variable Name	Description

Method: count_points

- **Single Responsibility:** Count scored points from a roll
- **Signature:** def count_points(roll_results)
- **Arguments:**
  - roll_results: List of results from a dice set roll.
- **Description:** Sums the points of all Dice values from roll_results unless the results contained a 5 or a 2.

Class Description: ScoreKeeper

Class Name:

ScoreKeeper

Single Responsibility:

Manages players' scores.

Instance Variables:

Variable Name	Description
scores	Hash set of all the player's cumulative scores from all their turns. Maps IDs to a scores

Method: get_player_scores

- **Single Responsibility:** Retrieve current player's scores
- **Signature:** def get_player_scores()
- **Arguments:** N/A
- **Description:** Returns a duplicate of the *scores* instance variable.

Method: update_points

- **Single Responsibility:** Update a specific player's score
- **Signature:** def update_points(player_id, points)
- **Arguments:**
  - player_id: ID of the player the request is updating the score for.
  - points: Number of additional points to add to the current player's score.
- **Description:** Once called, this method retrieves the player's score using its ID, adding the points argument to the score.

Class Description: Dice

Class Name:

Dice

Single Responsibility:

Simulate a dice being rolled

Instance Variables:

Variable Name	Description
sides	Number of sides of the dice

Method: initialize

- **Single Responsibility:** Initialize object instance state.
- **Signature:** def initialize(sides)
- **Arguments:**
  - sides: Number of sides of the dice
- **Description:** Sets *sides* instance variable to the sides argument.

Method: roll

- **Single Responsibility:** Roll the dice.
- **Signature:** def roll()
- **Arguments:** N/A
- **Description:** Once called, this method returns a random number between 1 and the number of sides inclusive.

Class Description: DiceSet

Class Name:

DiceSet

Single Responsibility:

Group multiple Dice instances together for the purpose of assigning them to a player.

Instance Variables:

Variable Name	Description
dice	List of dice instances that compose the Dice Set.

Method: initialize

- **Single Responsibility:** Initialize object instance state.
- **Signature:** def initialize(dice_count, sides)
- **Arguments:**
  - dice_count: Number of dice in the set.
  - sides: Number of sides of each dice
- **Description:** Creates a list of *dice_count* dice instances, where each dice has *sides* sides.

Method: roll_dice_set

- **Single Responsibility:** Roll all the dice in the dice set.
- **Signature:** def roll_dice_set()
- **Arguments:** N/A
- **Description:** Once called, this method calls each Dice's *roll* method and returns the results.

Class Description: DiceFactory

Class Name:

DiceFactory

Single Responsibility:

Create a DiceSet with pre-existing configurations.

Instance Variables:

Variable Name	Description
dice_count	Number of dice per dice set.
sides	Number of sides per dice.

Method: initialize

- **Single Responsibility:** Initialize object instance state.
- **Signature:** def initialize()
- **Arguments:** N/A
- **Description:** Initialize instance variables with default values.

Method: configure_dice

- **Single Responsibility:** Mutate the dice set creation configuration.
- **Signature:** def configure_dice(dice_count, sides)
- **Arguments:**
  - dice_count: Number of dice per dice set.
  - sides: Number of sides per dice.
- **Description:** Once called, this method mutates the instance variables with the values passed in to the arguments. These configurations are used for later creation of DiceSet instances.

Method: create_dice_set

- **Single Responsibility:** Create a dice set.
- **Signature:** def create_dice_set()
- **Arguments:** N/A
- **Description:** Once called, this method initializes a new *DiceSet* instance using configured *dice_count* and *sides.*

Class Description: AutoDropDead

Class Name:

AutoDropDead

Single Responsibility:

Automatically play the DropDead game / DropDead simulator client.

Instance Variables:

Variable Name	Description
drop_dead	DropDead API gateway object.
players	Players object that performs players' actions
arena	Arena object that initializes the game's state.
scoreboard	ScoreBoard object that announces the scores.
dice_set_controller	DiceSetController object that handles dice set configurations.

Method: initialize

- **Single Responsibility:** Initialize object instance state
- **Signature:** def initialize()
- **Arguments:** N/A
- **Description:** This method initializes all the instance variables listed above to new instances of their respective classes.

Method: play_game

- **Single Responsibility:** Plays the *DropDead* game.
- **Signature:** def play_game(sides, dice_count, player_count)
- **Arguments:**
  - sides: Number of sides per dice created
  - dice_count: Number of dice later created for each player's DiceSet
  - player_count: Number of players participating in the game.
- **Description:** Once the method is called, it initializes a new *Arena,* delegates the players' turns to *players* instance variable. Score is announced by the scoreboard instances variable.

Class Description: Arena

Class Name:

Arena

Single Responsibility:

Perform an AutoDropDead game configurations

Instance Variables:

Variable Name	Description

Method: initialize_game

- **Single Responsibility:** Performs an AutoDropDead game configurations
- **Signature:** def initialize_game(player_count, sides, dice_count, players, drop_dead, dice_set_controller)
- **Arguments:**
  - player_count: Number of players participating in the game.
  - sides: Number of sides per dice
  - dice_count: Number of dice per each player's dice set
  - players: Players object that performs players' actions
  - drop_dead: DropDead API gateway object
  - dice_set_controller: DiceSetController object that handles dice set configurations
- **Description:** Once the method is called, it initializes a *Players* instance with dice configurations.

Class Description: Players

Class Name:

Players

Single Responsibility:

Automate players' turns.

Instance Variables:

Variable Name	Description
players_controllers	List of PlayerController objects used to perform each player's turn.
alive_players	Number of remaining alive players.

Method: initialize

- **Single Responsibility:** Initialize object instance state.
- **Signature:** def initialize()
- **Arguments:** N/A
- **Description:** Once the method is called, it initializes instance variables listed above to their default empty values.

Method: initialize_players

- **Single Responsibility:** Populate instance variables with new game's simulation values.
- **Signature:** def initialize_players(player_count, drop_dead)
- **Arguments:**
  - player_count: Number of players in the new simulation.
  - drop_dead: DropDead object for DropDead API requests.
- **Description:** Once the method is called, it adds *player_count* number of PlayerController instances to *player_controllers,* mutating the number of *alive_players.*

Method: perform_turns

- **Single Responsibility:** Perform the players' turns simulation.
- **Signature:** def perform_turns()
- **Arguments:** N/A
- **Description:** Once the method is called, it performs all players' turns in a loop calling each player's controller *perform_turn* method until no more alive players remain.

Method: on_drop_dead

- **Single Responsibility:** Handle player's death (Callback from perform_turns)
- **Signature:** def on_drop_dead(player_id)
- **Arguments:**
  - player_id: The ID associated with the dead player.
- **Description:** Once the method is called, it announces the player's death, decrementing *alive_players.*

Method: on_roll

- **Single Responsibility:** Announce the player's roll. (Callback from perform_turns)
- **Signature:** def on_roll(roll_results, player_id)
- **Arguments:**
  - roll_results: Results from
  - player_id: The ID associated with the player who just rolled the dice.
- **Description:** Once the method is called, it announces the player's roll including the points scored from it.

Class Description: PlayerController

Class Name:

PlayerController

Single Responsibility:

Perform player's action requests.

Instance Variables:

Variable Name	Description
drop_dead	DropDead API gateway object.
player_id	Player's unique identifier.

Method: initialize

- **Single Responsibility:** Initialize object instance state
- **Signature:** def initialize()
- **Arguments:** N/A
- **Description:** This method initializes all the instance variables listed above to empty values.

Method: create_player

- **Single Responsibility:** Populate instance variables with new player's values
- **Signature:** def create_player(drop_dead)
- **Arguments:**
  - drop_dead: DropDead API gateway object for routing requests.
- **Description:** Once the method is called, it routes the request to *drop_dead*'s method *invite_player* populating the instance variables with ID and drop_dead argument.

Method: perform_turn

- **Single Responsibility:** Request player's turn from *drop_dead.*
- **Signature:** def perform_turn(on_drop_dead, on_roll)
- **Arguments:**
  - on_drop_dead: Callback to be called when the player dies.
  - on_roll: Callback to be called when a successful roll is performed.
- **Description:** Once the method is called, it routes the request to the *drop_dead* instance variable object by calling its *perform_turn* method using the player's ID. Appropriate callback is called using the result.

Method: get_status

- **Single Responsibility:** Retrieve player's status.
- **Signature:** def get_status()
- **Arguments:** N/A
- **Description:** Once the method is called, it routes the request to the *drop_dead* instance variable object by calling its *get_player_status* method using the player's ID.

Class Description: DiceSetController

Class Name:

DiceSetController

Single Responsibility:

Request to configure dice from DropDead.

Instance Variables:

Variable Name	Description

Method: configure_dice

- **Single Responsibility:** Request to configure dice from DropDead.
- **Signature:** def configure_dice(dice_count, sides, drop_dead)
- **Arguments:**
  - dice_count: Number of dice per dice set.
  - sides: Number of sides per dice.
  - drop_dead: DropDead API gateway object to make the request to.
- **Description:** Once called, this it requests dice configuration from drop_dead argument by calling its *configure_dice* method using dice configuration arguments.

Class Description: Scoreboard

Class Name:

Scoreboard

Single Responsibility:

Inform players of their scores.

Instance Variables:

Variable Name	Description
drop_dead	DropDead API gateway object to request player's scores.

Method: initialize

- **Single Responsibility:** Initialize object instance state
- **Signature:** def initialize()
- **Arguments:** N/A
- **Description:** This method initializes instance variables to empty defaults.

Method: announce_player_scores

- **Single Responsibility:** Announce all players' scores.
- **Signature:** def announce_player_scores(drop_dead)
- **Arguments:**
  - drop_dead: DropDead API gateway object to make the request to.
- **Description:** Once called, it requests from drop_dead to retrieve players' scores by calling its *get_player_scores* method. Scores are announced to the players.

Method: get_player_scores

- **Single Responsibility:** Retrieve players' scores.
- **Signature:** def get_player_scores()
- **Arguments:** N/A
- **Description:** Once called, it requests from *drop_dead* instance variable to retrieve players' scores by calling its *get_player_scores* method.