**Hong Kong Institute of Vocational Education Discipline of Information Technology**

**ITP4507 - Contemporary Topics in Software Engineering**

**Assignment**

|  |  |
| --- | --- |
| **Student No.** |  |
| **Student Name** |  |
| **Program Code** | **IT314105** |
| **Class** | **E & L** |

**Cntent**

[**Assumptions regarding the problem context** 3](#_Toc151757466)

[**Application design with class diagram** 5](#_Toc151757467)

[Main structure: 5](#_Toc151757468)

[Factory Pattern: 6](#_Toc151757469)

[Command Pattern: 9](#_Toc151757470)

[Memento Pattern: 11](#_Toc151757471)

[**Discussion and explanation of the application** 12](#_Toc151757472)

[Main structure: 12](#_Toc151757473)

[Factory Pattern 12](#_Toc151757474)

[Command Pattern: 13](#_Toc151757475)

[Memento Pattern: 13](#_Toc151757476)

[**User Guide** 14](#_Toc151757477)

[Introduction 14](#_Toc151757478)

[Getting Started 14](#_Toc151757479)

[Notes 15](#_Toc151757480)

[**Test Plan** 16](#_Toc151757481)

[Test Objectives 16](#_Toc151757482)

[Test Categories 16](#_Toc151757483)

[Test Execution Flow 17](#_Toc151757484)

[Expected Results 17](#_Toc151757485)

[**Test Case** 18](#_Toc151757486)

[Player Addition Test (TestCommandAddPlayer) 18](#_Toc151757487)

[Team Change Test (TestCommandChangeTeam) 20](#_Toc151757488)

[Team Creation Test (TestCommandCreateTeam) 22](#_Toc151757489)

[Player Deletion Test (TestCommandDeletePlayer) 24](#_Toc151757490)

[Player Information Modification Test (TestCommandModifyPlayer) 26](#_Toc151757491)

[Set Current Team Test (TestCommandSetCurrentTeam) 28](#_Toc151757492)

**Assumptions regarding the problem context**

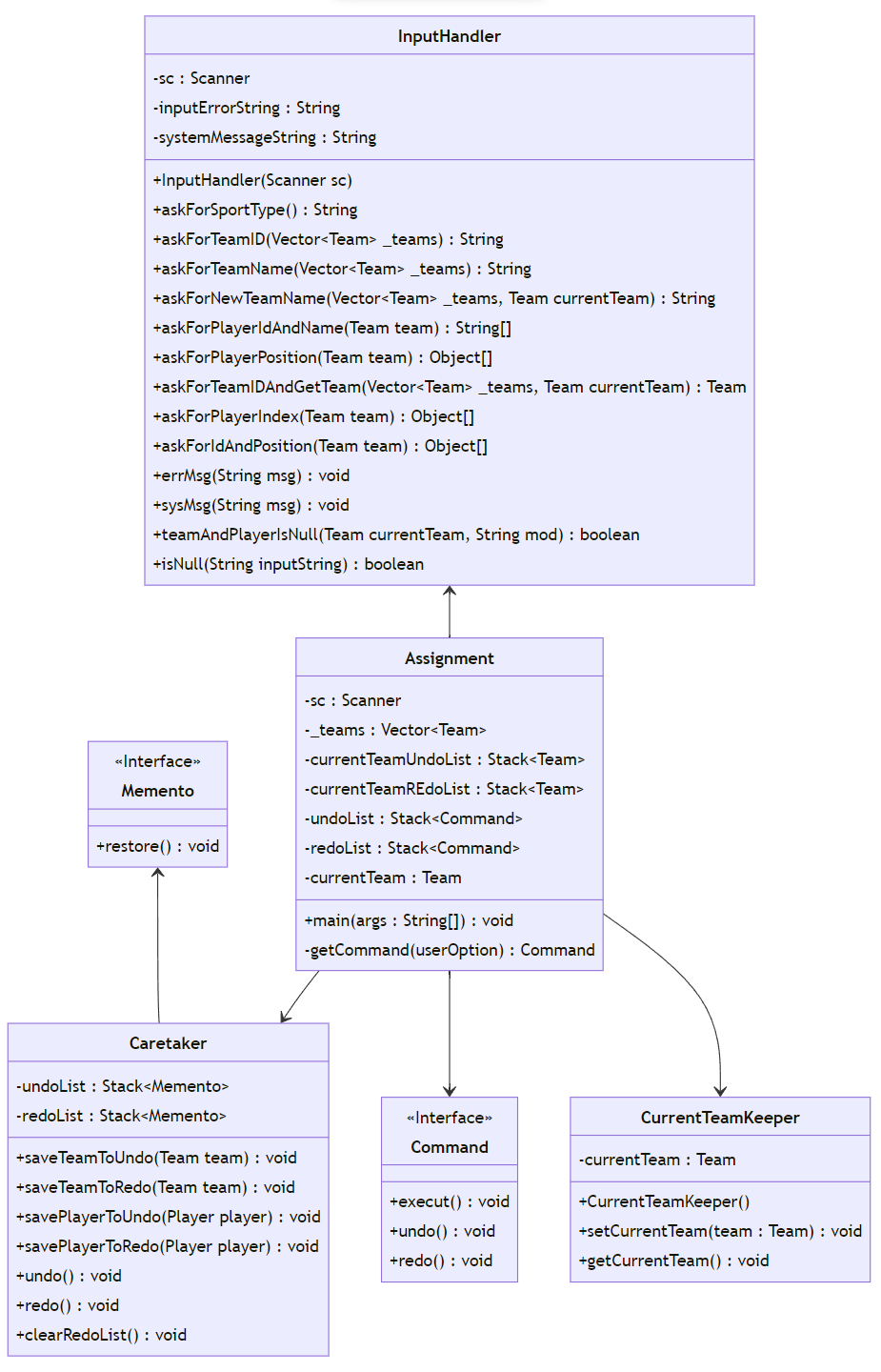
In the context of developing the Sport Teams Management System (STMS) for the Magical Athletes Teams Management Company, which currently supports volleyball and football and plans to expand to other team sports, the following assumptions are critical:

1. Diverse Sport Requirements: It is assumed that the needs and data structures of different sports vary significantly. Each sport, such as volleyball and football, has unique attributes and rules that the system must accommodate.
2. Dynamic Team and Player Information: The system is assumed to handle dynamic information regarding teams and players worldwide, requiring robust data management capabilities.
3. Command Pattern for User Commands: The STMS will utilize the Command Pattern for processing user commands. This design pattern is chosen with the assumption that it will efficiently handle user actions, such as adding, updating, or removing player information, by encapsulating them as objects.
4. Undo/Redo Functionality: An important assumption is the integration of undo and redo functionalities in the system. This feature is assumed to significantly enhance user error tolerance, allowing users to revert or reapply their actions easily. This is particularly crucial given the complexity and variability of sports team management.
5. Scalability for Future Expansion: The system is presumed to be scalable to accommodate additional team sports in the future. This includes the capability to integrate new sports with varying requirements without major overhauls to the existing system structure.
6. User Interface Adaptability: The user interface is assumed to adapt to different sports while maintaining a consistent user experience. This adaptability is crucial for ensuring that users can easily navigate and manage diverse sports teams and player information.

These assumptions form the foundation for the development and future expansion of the STMS, ensuring that it remains versatile, user-friendly, and capable of handling the complexities of managing various sports teams and players globally.

**Application design with class diagram**

Main structure:



Factory Pattern:

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 圖表 的圖片

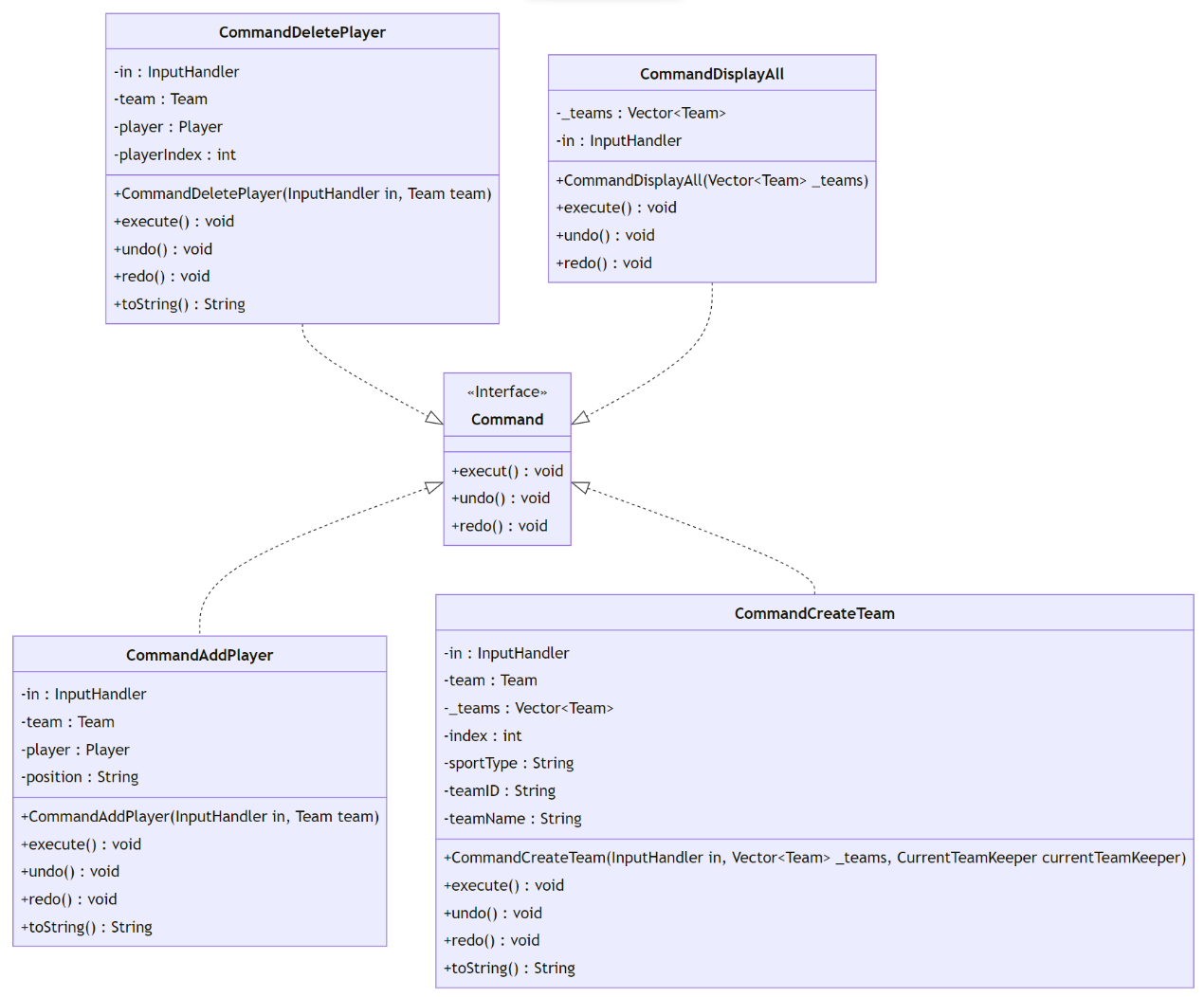
自動產生的描述

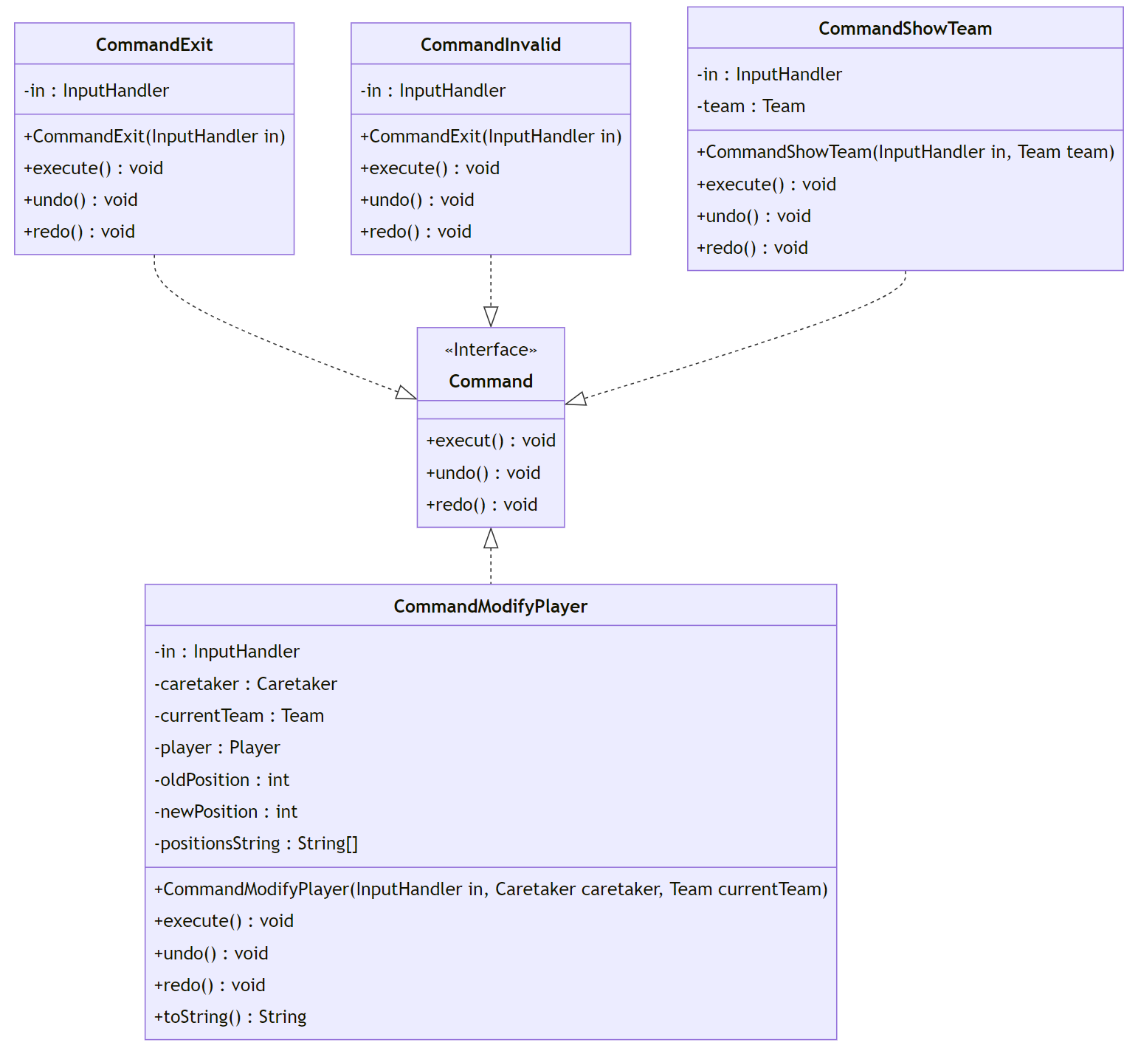
一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

Command Pattern:

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述 



一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

Memento Pattern:

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

**Discussion and explanation of the application**

Main structure:

In the Sport Teams Management System (STMS), the Command Pattern is used to handle user commands. All command objects are subclasses of the Interface Command, and are created through the getCommand() method. Any command that modifies the system data provides undo and redo functionality. Some of the undo and redo functionalities utilize the Memento Pattern. The InputHandler class is responsible for all functionalities related to user input and input validation.

Factory Pattern

The Factory Pattern is a type of creational design pattern that provides a way to create objects without specifying the exact class of object to be created. The main purpose of the Factory Pattern is to separate the creation of objects from their usage, making the code more flexible and easier to extend and maintain.

In the Factory Pattern, there is a class called the factory, which is responsible for creating objects based on the client's requirements. The client uses an interface provided by the factory to create objects without needing to instantiate concrete classes directly.

Benefits of the Factory Pattern include:

* Separating the creation and usage of objects, reducing coupling.
* Making it easier to extend and maintain the code by adding new products with corresponding concrete factories.
* Providing a centralized place for managing object creation, improving code organization and readability.

This system uses the Factory Pattern to create teams and players, and it also utilizes the Command Pattern to execute various operations, such as creating teams and adding players. This design approach enhances the system's extensibility and flexibility while enabling undoing and redoing operations.

Command Pattern:

The Command Pattern is a behavioral design pattern that encapsulates a request or simple operation as an object. This pattern allows users to separate the sender and receiver based on different requests and to perform operations on these requests, thereby enabling parameterization, queuing, or logging of requests. It enhances the flexibility and extensibility of commands.

In this scenario, each operation (such as creating a team or adding a player) is encapsulated within different command objects. Users execute specific actions by entering the correct command, allowing them to indirectly trigger these operations through the invocation of commands, rather than directly calling methods on the objects.

Memento Pattern:

The Memento Pattern is a behavioral design pattern that is used to capture and externalize the internal state of an object so that it can be restored to that state in the future. This pattern allows the object to be restored to a previous state without violating the encapsulation principle by not exposing its implementation details.

In this example, two Mementos are used: MementoTeam and MementoPlayer. These Mementos are employed to restore both the `Team` and `Player` objects to their previous states.

**User Guide**

Introduction

Sport Teams Management System (STMS) is a Java program for managing sports teams. It allows you to create teams, add players, modify player positions, delete players, change team names, and more. It also includes undo and redo functionality and the ability to list undo and redo operations.

Getting Started

1. Run the program: Execute the Java program, and you will enter STMS.
2. Main menu: Once the program is running, you will see the main menu of STMS. The main menu includes a list of available operations and the currently selected team.
3. Select an operation: Use the keyboard to input the corresponding command to perform the desired operation. Available operations include:

c: Create a new team.

g: Set the current working team.

a: Add a player to the current team.

m: Modify a player's position.

d: Delete a player.

t: Change the name of the current team.

s: Display detailed information about the current team.

p: List all teams.

u: Perform an undo operation.

r: Perform a redo operation.

l: List the history of undo and redo operations.

x: Exit the system.

1. Team Operations: When performing operations that require selecting a team (e.g., adding a player, modifying a player, deleting a player, changing the team's name), the system will prompt you to choose the current working team. Follow the prompts to complete the operation.
2. Undo and Redo: You can use the u and r commands to perform undo and redo operations, respectively. This allows you to backtrack and replay previous actions.
3. Listing Operation History: Use the l command to view the history of undo and redo operations.
4. Exiting the System: When you have finished your operations, use the x command to exit the system.

Notes

* Please input valid commands as prompted by the system; otherwise, the system will respond with "Invalid command."
* Ensure that you have selected the current working team when performing operations that require team selection.
* You can use the p command to list all teams and select the current working team from the list.
* Undo and redo operations apply to previous actions and have limitations based on the operation history.
* The system will prompt you for relevant information; please input the required information as prompted.

**Test Plan**

Test Objectives

The primary objective is to confirm the Sport Teams Management System (STMS) operates effectively, covering aspects like team management, player operations, and the undo/redo features. This includes validating data integrity, user interactions, and system responses under various scenarios.

Test Categories

1. Player Addition Test (TestCommandAddPlayer): Tests the player addition functionality, checking that new players can be correctly added to a team.
2. Team Change Test (TestCommandChangeTeam): Tests the team information change functionality, including modifications of team names.
3. Team Creation Test (TestCommandCreateTeam): Tests the team creation functionality, ensuring that new teams can be correctly added to the system.
4. Player Deletion Test (TestCommandDeletePlayer): Tests the player deletion functionality, ensuring players can be removed from a team.
5. Player Information Modification Test (TestCommandModifyPlayer): Tests the functionality for modifying player information, such as position changes.
6. Set Current Team Test (TestCommandSetCurrentTeam): Tests the functionality for setting the current team, ensuring the system can correctly identify and operate with the selected team.

Each test category thoroughly examines its respective functionality. Except for the "Set Current Team" test, which includes only the testExecute() method, the other tests should encompass testExecute(), testUndo(), testRedo(), and testToString() methods to ensure the system's functionalities are complete and reliable.

Test Execution Flow

1. In each test category, create corresponding JUnit test methods.
2. In each test method, use JUnit assertions to verify that the results of the operations match the expected outcomes.
3. Ensure that each test method is independent and does not rely on others.

Expected Results

* All test cases should pass, and the system's functionalities should operate as expected.
* Tests should run and pass without causing system errors or exceptions.

**Test Case**

Player Addition Test (TestCommandAddPlayer)

Test data – 10 items

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | 1 | 2 | 3 | 4 | 5 |
| playerID | P004 | P102 | P233 | 332 | @123 |
| playerName | Jacqueline Ramirez | Charles Watts | Kevin Perez | Michael Powell | Lisa Vargas |
| Case | 6 | 7 | 8 | 9 | 10 |
| playerID | 91966 | WDGV8 | WMI2X | EHR6X | XP72C |
| playerName | Nancy Davis | Cassandra Alexander | Sherry Nguyen | Daniel Villarreal | Michael Hurley |

Test result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | testExecute() | testUndo() | testRedo() | testToString() |
| 1 | PASS | PASS | PASS | PASS |
| 2 | PASS | PASS | PASS | PASS |
| 3 | PASS | PASS | PASS | PASS |
| 4 | PASS | PASS | PASS | PASS |
| 5 | PASS | PASS | PASS | PASS |
| 6 | PASS | PASS | PASS | PASS |
| 7 | PASS | PASS | PASS | PASS |
| 8 | PASS | PASS | PASS | PASS |
| 9 | PASS | PASS | PASS | PASS |
| 10 | PASS | PASS | PASS | PASS |

Test Methods

* testExecute(): Tests the logic of adding a player. It iterates through player IDs and names, sets the player's position, executes the command to add the player, and verifies if the results are correct.
* testUndo(): Tests the functionality of undoing the addition of a player. After adding players, it performs an undo operation and checks whether the players have been correctly removed from the team.
* testRedo(): Tests the redo functionality of adding a player. Following the undo operations, redo is executed to verify if players are correctly re-added.
* testToString(): Tests if the command's string representation is correct.

Helper method

* getLastPlayer(): Gets the last player from the team, used to verify test results.

Inner Class InputHandlerStub

* Used to simulate user input, providing methods to set player IDs, names, and positions.
* Overrides the methods for asking player ID, name, and position to be used in testing scenarios.

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Team Change Test (TestCommandChangeTeam)

Test data – 10 items

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | 1 | 2 | 3 | 4 | 5 |
| teamNames | T002 | GG | 8888 | F# | v8 |
| Case | 6 | 7 | 8 | 9 | 10 |
| teamNames | KFC | 6464! | >.< | DDDDFS | @#$@# |

Test result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | testExecute() | testUndo() | testRedo() | testToString() |
| 1 | PASS | PASS | PASS | PASS |
| 2 | PASS | PASS | PASS | PASS |
| 3 | PASS | PASS | PASS | PASS |
| 4 | PASS | PASS | PASS | PASS |
| 5 | PASS | PASS | PASS | PASS |
| 6 | PASS | PASS | PASS | PASS |
| 7 | PASS | PASS | PASS | PASS |
| 8 | PASS | PASS | PASS | PASS |
| 9 | PASS | PASS | PASS | PASS |
| 10 | PASS | PASS | PASS | PASS |

Test Methods

* testExecute(): Tests the execution of the CommandChangeTeam. It iterates through the teamNames array, sets each as the new team name, executes the command, and asserts that the team's name is correctly updated.
* testUndo(): Tests the undo functionality of the command. It first executes the command for each name and then undoes each change, verifying that the team's name reverts to its previous state.
* testRedo(): After undoing the changes, this test verifies the redo functionality, ensuring that the team's name is set back to its changed state.
* testToString(): Tests the string representation of the command, ensuring it correctly reflects the action taken.

Inner Class: InputHandlerStub

* A stub for the InputHandler class, overriding the method to return a new team name. This stub is used to simulate the user input that would be required in a real scenario.

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Team Creation Test (TestCommandCreateTeam)

Test data – 10 items

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | 1 | 2 | 3 | 4 | 5 |
| teamNames | T002 | GG | 8888 | F# | v8 |
| teamID | ZBB | 333 | @3 | GG | T999 |
| sportType | Volleyball | Football | Volleyball | Football | Volleyball |
| Case | 6 | 7 | 8 | 9 | 10 |
| teamNames | KFC | 6464! | >.< | DDDDFS | @#$@# |
| teamID | 7.7 | ##$$$& | 22ddff | #FFFFFF | :::: |
| sportType | Football | Volleyball | Football | Volleyball | Football |

Test result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | testExecute() | testUndo() | testRedo() | testToString() |
| 1 | PASS | PASS | PASS | PASS |
| 2 | PASS | PASS | PASS | PASS |
| 3 | PASS | PASS | PASS | PASS |
| 4 | PASS | PASS | PASS | PASS |
| 5 | PASS | PASS | PASS | PASS |
| 6 | PASS | PASS | PASS | PASS |
| 7 | PASS | PASS | PASS | PASS |
| 8 | PASS | PASS | PASS | PASS |
| 9 | PASS | PASS | PASS | PASS |
| 10 | PASS | PASS | PASS | PASS |

Test Methods

* testExecute(): Tests the execution of creating teams. It iterates through team names and IDs, sets up the team attributes, executes the command, and asserts that the teams are correctly added with the expected attributes.
* testUndo(): Tests the undo functionality. After creating teams, it undoes the creation of each team and verifies if the team is removed from the vector.
* testRedo(): Tests the redo functionality. It redoes the creation of teams after undo operations and verifies if the teams are added back correctly.
* testToString(): Tests the string representation of the command, ensuring it accurately reflects the create team action.

Helper Method

* isTeamExist(): Checks if a specific team exists in the vector.

Inner Class: InputHandlerStub

* A stub for the InputHandler class, providing methods to set team attributes and overriding methods to return these attributes. This stub simulates user input for team creation in the tests.

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Player Deletion Test (TestCommandDeletePlayer)

Test data – 10 items

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | 1 | 2 | 3 | 4 | 5 |
| playerID | P004 | P102 | P233 | 332 | @123 |
| playerName | Jacqueline Ramirez | Charles Watts | Kevin Perez | Michael Powell | Lisa Vargas |
| Case | 6 | 7 | 8 | 9 | 10 |
| playerID | 91966 | WDGV8 | WMI2X | EHR6X | XP72C |
| playerName | Nancy Davis | Cassandra Alexander | Sherry Nguyen | Daniel Villarreal | Michael Hurley |

Test result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | testExecute() | testUndo() | testRedo() | testToString() |
| 1 | PASS | PASS | PASS | PASS |
| 2 | PASS | PASS | PASS | PASS |
| 3 | PASS | PASS | PASS | PASS |
| 4 | PASS | PASS | PASS | PASS |
| 5 | PASS | PASS | PASS | PASS |
| 6 | PASS | PASS | PASS | PASS |
| 7 | PASS | PASS | PASS | PASS |
| 8 | PASS | PASS | PASS | PASS |
| 9 | PASS | PASS | PASS | PASS |
| 10 | PASS | PASS | PASS | PASS |

Test Methods

* testExecute(): Tests the execution of deleting a player. It iterates through players, sets each one for deletion, executes the command, and checks if the player is successfully deleted from the team.
* testUndo(): Tests the undo functionality. After deleting players, it undoes each deletion and checks if the players are correctly added back to the team.
* testRedo(): Tests the redo functionality. It redoes the deletion of players after undo operations and verifies if the players are removed again as expected.
* testToString(): Tests the string representation of the command, ensuring it accurately reflects the delete action.

Helper Methods

* getTeamPlayerLength(): Returns the number of players in the team.
* getPlayer(Enumeration<Player> players, Player player): Returns a player from the enumeration if it matches the specified player.
* getPlayer(Enumeration<Player> players, int index): Returns a player at the specified index in the enumeration.

Inner Class: InputHandlerStub

* A stub for the InputHandler class, providing a method to set the player and override a method to return the index and player object. This stub simulates user input in the test environment.

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Player Information Modification Test (TestCommandModifyPlayer)

Test data – 10 items

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | 1 | 2 | 3 | 4 | 5 |
| playerID | P004 | P102 | P233 | 332 | @123 |
| playerName | Jacqueline Ramirez | Charles Watts | Kevin Perez | Michael Powell | Lisa Vargas |
| Case | 6 | 7 | 8 | 9 | 10 |
| playerID | 91966 | WDGV8 | WMI2X | EHR6X | XP72C |
| playerName | Nancy Davis | Cassandra Alexander | Sherry Nguyen | Daniel Villarreal | Michael Hurley |

Test result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | testExecute() | testUndo() | testRedo() | testToString() |
| 1 | PASS | PASS | PASS | PASS |
| 2 | PASS | PASS | PASS | PASS |
| 3 | PASS | PASS | PASS | PASS |
| 4 | PASS | PASS | PASS | PASS |
| 5 | PASS | PASS | PASS | PASS |
| 6 | PASS | PASS | PASS | PASS |
| 7 | PASS | PASS | PASS | PASS |
| 8 | PASS | PASS | PASS | PASS |
| 9 | PASS | PASS | PASS | PASS |
| 10 | PASS | PASS | PASS | PASS |

Test Methods

* testExecute(): Tests the execution of modifying a player's position. It iterates through players, sets each one's new position, executes the command, and checks if the player's position is successfully updated.
* testUndo(): Tests the undo functionality. After modifying players' positions, it undoes each modification and verifies if the players' positions revert to their original state.
* testRedo(): Tests the redo functionality. It redoes the position modifications after undo operations and verifies if the players' positions are changed back as expected.
* testToString(): Tests the string representation of the command, ensuring it accurately reflects the modify action.

Helper Method

* getPlayer(Enumeration<Player> players, int index): Retrieves a player at the specified index from the enumeration.

Inner Class: InputHandlerStub

* A stub for the InputHandler class, providing methods to set the player and position and override a method to return player and position. This stub simulates user input in the test environment.

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Set Current Team Test (TestCommandSetCurrentTeam)

Test data – 10 items

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | 1 | 2 | 3 | 4 | 5 |
| teamNames | T002 | GG | 8888 | F# | v8 |
| teamID | ZBB | 333 | @3 | GG | T999 |
| Case | 6 | 7 | 8 | 9 | 10 |
| teamNames | KFC | 6464! | >.< | DDDDFS | @#$@# |
| teamID | 7.7 | ##$$$& | 22ddff | #FFFFFF | :::: |

Test result

|  |  |
| --- | --- |
| Case | testExecute() |
| 1 | PASS |
| 2 | PASS |
| 3 | PASS |
| 4 | PASS |
| 5 | PASS |
| 6 | PASS |
| 7 | PASS |
| 8 | PASS |
| 9 | PASS |
| 10 | PASS |

Test Method

* testExecute(): Tests the execution of setting the current team. It first checks that no team is set as current. Then it adds teams to the vector and tests setting each team as the current team randomly 20 times, verifying that the current team is correctly updated each time.

Inner Class: InputHandlerStub

* A stub for the InputHandler class, providing a method to set a team and overriding a method to return a team. This stub simulates user input for selecting a team in the tests.

一張含有 字型, 文字, 圖形, 螢幕擷取畫面 的圖片

自動產生的描述