Tools API

Token

Source File: Tools.h
Namespace: ds
Class Header: `class Token : public Object`

Overview

The *Token* class constructs a constant comparable string-integer pair with read-only access to the key and the ability to hide the key and temporarily alter the value.

Constructors

- `Token()` (default constructor)
  - **Purpose:** Creates a key-value pair whose key is the empty string and whose value is zero.
- `Token(const Token& obj)` (copy constructor)
  - **Purpose:** Constructs a deep copy of *obj*.
  - **Parameter(s):**
    - *obj*: Constant *Token* reference object.
- `Token(string key, int value)`
  - **Purpose:** Creates a key-value pair whose key is *key* and whose value is *value*.
  - **Parameter(s):**
    - *key*: The content of the key.
    - *value*: The content of the value.

Destructor

- `~Token()`
  - **Purpose:** Does nothing.

Assignment Operators

- `operator=(const Token& rhs)`
  - **Purpose:** Constructs a deep copy of *rhs*.
  - **Parameter(s):**
    - *rhs*: Constant *Token* reference object.
  - **Return:** `*this`.

Methods

- `Hide(string value)`
  - **Purpose:** Hides the key with *value* in the display of the object.
  - **Parameter(s):**
    - *value*: A cover for the key.
- `Suspend(unsigned int value)`
  - **Purpose:** Sets the numbers of consecutive comparisons that can use an alternative value of the value.
  - **Parameter(s):**
    - *value*: A count.
- `Alter(int value)`
  - **Purpose:** Assigns a temporary value to value.
  - **Parameter(s):**
    - *value*: A temporary value.
- `IsHidden() const`
  - **Purpose:** Checks if the key is hidden.
  - **Return:** True if the key is hidden; otherwise, false.

- `Key() const`
    - **Purpose:** Retrieves the key.
    - **Return:** The key.
- `Reveal()`
    - **Purpose:** Unhides the key.
- `Restore()`
    - **Purpose:** Restores the value.
- `ToString() const override`
    - **Purpose:** Provides a string representation of the key.
    - **Return:** The key if it is not hidden; otherwise, the cover of the key.

Non-Member Functions

- `operator==(const Token& lhs,const Object& rhs)`
    - **Purpose:** Checks if the values of *lhs* and *rhs* are equal.
    - **Parameters:**
        - *lhs*: Constant reference of an *Token* object.
        - *rhs*: Constant reference of an *Token* object.
    - **Return:** True if their values are equal; otherwise, false.
- `operator!=(const Token& lhs,const Object& rhs)`
    - **Purpose:** Checks if the values of *lhs* and *rhs* are different.
    - **Parameters:**
        - *lhs*: Constant reference of an *Token* object.
        - *rhs*: Constant reference of an *Token* object.
    - **Return:** True if their values are different; otherwise, false.
- `operator<(const Token& lhs,const Object& rhs)`
    - **Purpose:** Checks if the value of *lhs* is less than the value of *rhs*.
    - **Parameters:**
        - *lhs*: Constant reference of an *Token* object.
        - *rhs*: Constant reference of an *Token* object.
    - **Return:** True if *lhs*'s values is less than *rhs*'s value; otherwise, false.
- `operator>(const Token& lhs,const Object& rhs)`
    - **Purpose:** Checks if the value of *lhs* is greater than the value of *rhs*.
    - **Parameters:**
        - *lhs*: Constant reference of an *Token* object.
        - *rhs*: Constant reference of an *Token* object.
    - **Return:** True if *lhs*'s values is greater than *rhs*'s value; otherwise, false.
- `operator<=(const Token& lhs,const Object& rhs)`
    - **Purpose:** Checks if the value of *lhs* is less than or equal to the value of *rhs*.
    - **Parameters:**
        - *lhs*: Constant reference of an *Token* object.
        - *rhs*: Constant reference of an *Token* object.
    - **Return:** True if *lhs*'s values is less than or equal to *rhs*'s value; otherwise, false.
- `operator>=(const Token& lhs,const Object& rhs)`
    - **Purpose:** Checks if the value of *lhs* is greater than or equal to the value of *rhs*.
    - **Parameters:**
        - *lhs*: Constant reference of an *Token* object.
        - *rhs*: Constant reference of an *Token* object.
    - **Return:** True if *lhs*'s values is greater than or equal to *rhs*'s value; otherwise, false.
- `operator-(const Token& lhs,const Object& rhs)`
    - **Purpose:** Retrieves the distance between the values of *lhs* and *rhs*.
    - **Parameters:**
        - *lhs*: Constant reference of an *Token* object.
        - *rhs*: Constant reference of an *Token* object.
    - **Return:** The absolute value of the difference of the values of *lhs* and *rhs*.

Collection

Source File: Tools.h
Namespace: ds
Class Header: `class Collection : public Object`

Overview

The *Collection* class is an interface class designed to serve a container of *Token* objects.

Member Functions

- `Insert(const Token& value, string flag)`
  - **Purpose:** Intended to add *value* to the collection based on *flag*.
  - **Parameters:**
    - *value*: Constant reference of an *Token* object.
    - *flag*: Determines the action.
- `Remove(string flag)`
  - **Purpose:** Intended to remove a token from the collection based on *flag*.
  - **Parameters:**
    - *flag*: Determines the action.
- `View(string flag)`
  `View(string flag) const`
  - **Purpose:** Intended to retrieve a token from the collection based on *flag*.
  - **Parameters:**
    - *flag*: Determines the action.
  - **Return:** A reference of a *Token* object.
- `Extend(bool value)`
  - **Purpose:** Intended to set the collection display.
  - **Parameters:**
    - *value*: Sets the display mode.
- `Size() const`
  - **Purpose:** Intended to retrieve the collection size.
  - **Return:** An integer.
- `Empty() const`
  - **Purpose:** Intended to check if the collection is empty.
  - **Return:** True if the collection is empty; otherwise, false.
- `Clear()`
  - **Purpose:** Intended to empty the collection.
- `Info() const`
  - **Purpose:** Intended to provide a list of all the flags.
  - **Return:** A string.
- `ToString() const override`
  - **Purpose:** Pure virtual function to be implemented by derived classes.
  - **Return:** A string representation of the collection.

Program

Source File: Tools.h
Namespace: ds
Class Header: `class Program : public Object`

Overview

The *Program* class is an interface class designed to maneuver tokens of *Collection* objects with undo capabilities.

Member Functions

- `Initialize()`
    - **Purpose:** Intended to add initialize the program.
- `Move(string flag)`
    - **Purpose:** Intended to make a move in the program based on *flag*.
    - **Parameters:**
        - *flag*: Determines the action.
- `Undo()`
    - **Purpose:** Intended to undo previous moves.
    - **Return:** True if an undo was successful; otherwise, false.
- `Completed() const`
    - **Purpose:** Intended to check if the program has no more moves.
    - **Return:** True if no moves are possible; otherwise, false.
- `State() const`
    - **Purpose:** Intended to retrieve the program's status.
    - **Return:** A string.
- `Info() const`
    - **Purpose:** Intended to provide a list of all the flags.
    - **Return:** A string.
- `ToString() const override`
    - **Purpose:** Pure virtual function to be implemented by derived classes.
    - **Return:** A string representation of the program.