Iain Black

CS-273

Dr. Bell

Assignment #3a

**Requirement Specification**

For the Crawlspace program, the general idea is that there is the user moves through a series of locations, at each of which they will have the option to examine the location, pick up what is in the location, or move to a different location. Users should have the ability to check the location that they are in for treasure, and then pick up that item, which will then be stored in their inventory. By Upon selecting the option to examine, a description of what they choose to examine will be displayed, they then have the option to pick it up. Upon choosing to move, a list of all available exits will be displayed for the user to choose from.

**Use Cases**

Sub-problems:

* Examine current location
* Take treasures in location and add to inventory
* Move locations

Examine

|  |  |  |
| --- | --- | --- |
| Step | **User Action** | **System Response** |
| 1. | User selects “examine” option. |  |
| 2. |  | System prompts the user to select what they would like to examine, displaying a list containing a “Room” option, objects within the location, and a “Done” option. |
| 3. | User selects “Room”. |  |
| 4. |  | System displays description of the location then returns to list. |
| 5. | User selects an object. |  |
| 6. |  | System displays description of the object then returns to list. |
| 7. | User selects “Done”. |  |
| 8. |  | System jumps back to beginning of game loop, displays original three options. |

Take

|  |  |  |
| --- | --- | --- |
| Step | **User Action** | **System Response** |
| 1. | User selects “Take” option. |  |
| 2. |  | System prompts user to enter item they wish to take, or enter “done” to exit. |
| 3. | User enters name of object that is not within the location. |  |
| 4. |  | System displays “That’s not here” and then prompts the user to enter the name of another item. |
| 5. | User enters name of a treasure within the location. |  |
| 6. |  | System displays “You got <N> points”, where <N> is the value of the treasure and then prompts the user to enter the name of another item. Treasure is added into explorer’s inventory. |
| 7. | User enters name of an exit within the location. |  |
| 8. |  | System displays “What a concept!” and then prompts the user to enter the name of another item. |
| 9. | User enters “Done”. |  |
| 10. |  | System jumps back to beginning of game loop, displays original three options. |

Move

|  |  |  |
| --- | --- | --- |
| Step | **User Action** | **System Response** |
| 1. | User selects “move” option. |  |
| 2. |  | System prompts the user to select which exit they would like to take, displays list of exits. |
| 3. | User selects exit not within the list |  |
| 4. |  | System displays “You can’t move that way!” and returns to list. |
| 5. | User selects exit within list |  |
| 6. |  | System updates current location of explorer, begins game loop. |

**UML**

|  |
| --- |
| **Crawlspace** |
| -vector<Location\*> maze  -unsigned short start  -unsigned short size  -int current\_location |
| **+**Crawlspace()  +void addLocation(Location\*)  +void updateLoc(Exit&, int)  +void updateLoc(Treasure&, int)  +void setStart(unsigned short, unsigned short)  +void setCurrentLocation(int)  +Location\* getCurLoc() |

|  |
| --- |
| **Object** |
| -string name  -string descr  -int value |
| **+**Object()  +void describeme()  +string getName()  +int getValue()  +virtual Object\* takeme() =0; |

|  |
| --- |
| **Location** |
| -string name  -string descr  -list<Exit> exits  -list<Treasure> inventory |
| **+**Location()  +void dig(Exit)  +void drop(Treasure)  +int exitLocation(string name)  +Object\* find(string name)  +void printName()  +void listExits()  +void listTreasures()  +void searchExits(string)  +void searchTreasures(string)  +void describe() |

|  |
| --- |
| **Exit** |
|  |
| +Exit()  +Object\* takeme() |

|  |
| --- |
| **Treasure** |
|  |
| **+**Treasure()  + Object\* takeme() |

**Psuedo-Code**

Location class:

**Function dig**

Function dig is used to add an exit to a given location’s list of exits, by passing in an exit object the function uses push\_back to add the exit to the list exits.

**Algorithm for dig**

1. Reads name of exit object
2. Uses push\_back to add exit to list exits.

**Function drop**

Function drop is used to add a treasure to the current location’s list of treasures, by passing in a treasure object the function uses push\_back to add the treasure to the list inventory.

**Algorithm for drop**

1. Reads name of treasure object
2. Uses push\_back to add treasure to list inventory.

**Function exitLocation**

Function exitLocation is used to search a given location for an exit by name, returning the associated location number from the list exits.

**Algorithm for exitLocation**

1. Reads in string for the name of desired exit
2. Sets iterator cur to the first member of list exits
3. *While* (cur doesn’t equal the last member of the list)
4. *If* (the name of the current member equals that of the read string)
5. Return the value of cur using getValue
6. Increment cur
7. If no members’ names match the read string return -1

**Function find**

Function find is used to search a given location for a treasure by name, returning the object and deleting it from the list inventory.

**Algorithm for find**

1. Reads in string for name of desired treasure
2. Create null pointer of type object called item
3. Sets iterator cur to first member of list inventory
4. *While* (cur doesn’t equal the last member of the list)  
    *If* (the name of the current member equals that of the read string)  
    call takeme function and set it to item
5. Erase cur from inventory
6. Break to jump out of loop
7. *If* (item is still null)
8. Throw an exception stating that the treasure was not found
9. Return item

**Function describe**

Function describe is used to display the name and written description of a given location as well as list any exits and treasures that it contains.

**Algorithm for describe**

1. Displays location’s private variable name
2. Displays location’s private variable descry
3. *If* (list exits is not empty)
4. Call listExits function
5. *If* (list Inventory is not empty)
6. Call listTreasures function.