Chapter 8 Book Work

Logan Yeubanks

8.1:

1. Text based adventure game that seems to revolve around getting out of a cave.

2. help, quit, go, go east, go west, go north, go south

3. quit exits the game, the go commands move the player, and the help command gives information about the game.

4. 4 rooms

8.2:

1. command class validates each command

2. game class is the main class and

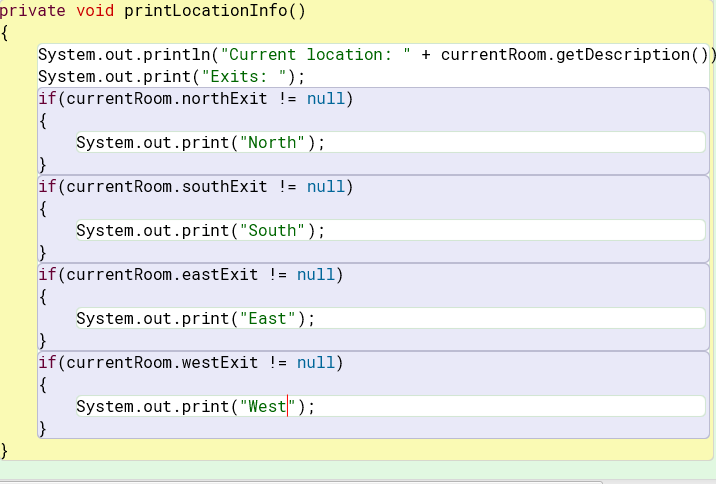
3. room class gives the description and exit for each room

4. commandworlds class stores all commands that have been given and shows all valid commands

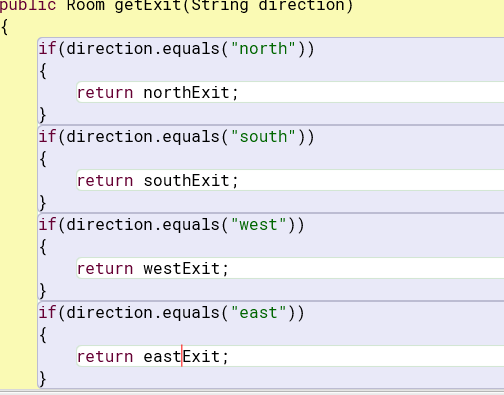
5. parses the players commands

8.3: A adventurer searching through a cave for the fountain of youth.

8.5:



8.6:

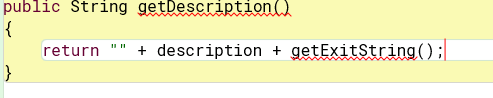




8.9: The keyset method returns the keys inside a HashMap.

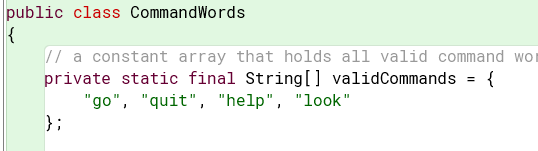
8.10: the getExitString method returns the exits in the current room. It is a Hashmap with all keys to exits. It then creates a string from that and returns it.

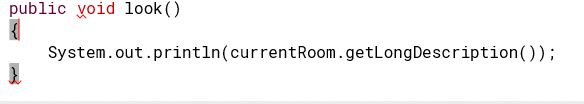
8.11:



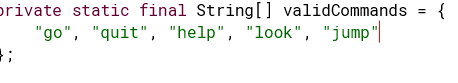
8.13: The starting location is reset to a new location.

8.14:





8.15:



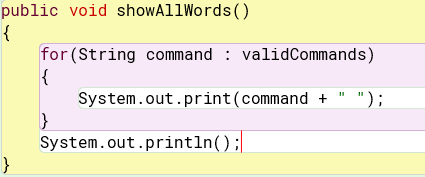
Else if (commandWord.equals(“jump”))

{

System.out.println(“You jumped!”);

}

8.16:



8.17: Yes you would need to change it in the Game class because the commands are only ran inside the Game class.

8.19: The model view controller pattern gives the program a user interface.

8.21: I think the room class should describe the item because the item is inside of the room.

8.24: The back command worked how it should.

8.25: It keeps the player in the same room making it invalid.

8.26: I would test that all directional commands go to where they are supposed to and that rooms were in the right places.

8.30:

public String[] itemCarry = new String[20];

public int I = 0;

public void pickUpItem() {

this.itemCarry[i] = itemName;

System.out.println(“Carrying: “ + itemCarry[i]);

I++

}

public void dropItem() {

if(I <= 0) {

System.out.println(“No items available to drop”);

}

Else {

--I;

System.out.println(“Dropped: “ + itemCarry[i]);

this.itemCarry[i] = null

}

}

8.31:

Public int maxWeight = 0;

public void pickUpItem() {

if((maxWeight + weight) < 1000) {

this.itemCarry[i] = itemName;

System.out.println(“Picked up: “ + itemCarry[i]);

I++;

maxWeight += weight;

}

Else {

System.out.println(“Item is too heavy to carry”);

}

}

8.32:

public void printItem() {

System.out.println(“Current items are: “);

For(int w = 0; w < I; w++) {

System.out.println(“ “ + (w + 1) + “ “ + itemCarry[w]);

}

}

8.35:

public void look() {

System.out.println(currentRoom.getDescription());

}

8.36: Yes the commandWorlds class needs to be changed to make this work.

8.37: When I do this change the welcome message stays the same.

8.38:

public enum Direction

{

NORTH, SOUTH, EAST, WEST

}

8.40: This change did reflect in the program.

8.41:

boolean done = false;

while (!done) {

if(counter > 10) {

System.out.println(“You have ran out of time”);

Done = true;

}

}