# Project 5: Zork - Text Adventure Game Using Arrays

Using arrays in Java within Replit.com (or any other IDE that allows students to work together on code), students will work in pairs to create a text-based adventure game!

## Overview

This game takes place in a three-story dungeon.  The user has to traverse the levels in search of the prize.  Along the way they collect items and fight monsters. On each move the user has seven possible commands: `left`, `right`, `up`, `down`, `grab`, `fight`, and `help` .  If the input is invalid (not one of these commands), the game will let the user know.  Otherwise, the game will execute the user's command.  The goal of the game is to collect the prize guarded by the boss monster.

# Details

## Example Game Play

```
What would you like to do? left
You see a sword.
What would you like to do? grab
you picked up the sword.
What would you like to do? left
You see nothing.
What would you like to do? left
You see a monster.
What would you like to do? fight
You defeated the monster!
What would you like to do? left
You see stairs.
What would you like to do? down
You see nothing.
What would you like to do? right
You see stairs.
What would you like to do? left
You see nothing.
What would you like to do? left
Sorry you can't go that way.
What would you like to do? up
You cannot go that way.
What would you like to do? right
You see nothing.
What would you like to do? up
You cannot go that way.
```

```
What would you like to do? right
You see stairs.
What would you like to do? up
You see nothing.
What would you like to do? right
You see nothing.
What would you like to do? right
You see nothing.
What would you like to do? right
You see nothing.
What would you like to do? up
You cannot go that way.
What would you like to do? left
You see nothing.
What would you like to do?
```

**The game has three floors**

- Each floor is made up of five (5) or more rooms, arranged in a line from left to right.
- A room can contain: a sword, a monster, magic stones, stairs, or nothing.

**At the start of the game, the user is placed in one of the rooms.**

**Movement**

- The user can try to move to the left room or right room.
- If there is no room in that direction, the game should report this.
- The user can also move upstairs or downstairs only if the room contains stairs.

**Contents of rooms**

- The game prints out the contents of the current room after every command.
- The user can grab swords or magic stones if they walk into a room containing either of these items.
- The sword or stones are no longer in the room once grabbed.

**Monsters guard some rooms**

- The user can use a sword to defeat a monster using the `fight` command. The sword and monster disappear after fighting.
- If they have no sword, the user can exit in the direction from which they came.
- If the user fights without a sword, they will be defeated and the game will end.
- If they try to walk past a monster, they will be killed and the game will end.
- Both a sword and magic stones are required to defeat the boss monster.

# Implementation Details

You will need to include pre-programming in the form of pseudocode.

1. Determine what input your program will prompt the user for.
2. Determine how your conditionals, including Boolean logic structure, will basically work.
3. Decide what roll each function will take.
4. Decide how your array(s) will be organized.

The following **must** be included in the program:

- Create procedures and functions (unlike procedures your two (2) or more functions return variables) for all options a player can take, but in Java these are both methods.
- Include error handling of user input.
- Include Boolean logic with one or more sets nested using && (and) and/or || (or).
- Provide meaningful and potentially useful comments (//).
- Include random number generation to determine the outcome of some major events such as combat.
- Use a switch statement for the various commands.
- Use a two-dimensional (2-D) array(s). (See the Bonus for three-dimensional array(s).)

### User Items

- Use an array to keep track of the user's items.
- At the beginning of the game it should be empty.
- A maximum of three items can be held at once.

### Monsters

- There should be three regular monsters placed throughout the game.
- These monsters require a sword to defeat.
- There should be a boss monster in the room just before the room that contains the prize.
- This monster requires both magic stones and a sword to defeat.

### Movement Implementation

- The user can only go up if there is a staircase.
- The program should not allow the user to run past a monster. (See the Bonus for exceptions.)
- The program should not allow the user to go up, down, or past bounds of the game.

### Win/Lose

- The game is won when the player grabs the prize.
- The game is lost if the user:
  fights a monster without a sword
  fights the boss monster without a sword and stones or tries to move past a monster.

# Design Considerations

### Game Board

The game board is the basis of this game. The following is a way to think of a smaller game board as a two-dimensional (2-D) array comprised of rooms on floors.

```
String[][] floorArray =  {{ "nothing", "nothing", "stairs" },
                          { "nothing", 'nothing', "stairs" },
                          { "prize", "nothing", "nothing" }} ;
```

### User Position

It will be useful to keep track of the user's position through multiple variables (or a 1-D array).

```
int userRoom = 1
int userFloor = 1
```

The above code  would put the user at the position of the first room of the first floor.

### Validating User Input

You will need to check the input of the user to make sure it is valid for the current game state:

```
if(userInput == "down"){
   if(floorArray[1][1] != "stairs"){
      System.out.println("You can't go downstairs; there are no stairs in this room.);
   }
}
```

**Submit your and pseudocode (pre-planning) via Schoology as an MS Word doc (*.docx).**

**Submit your complete Java program via Schoology.**

## Bonus (up to 100 A+)

- Add the command run, which allows a player to run past a monster instead of fighting.
  This should work less than 50% of the time.
- At the start of the game, place the user in one of the rooms at random.
  It is recommended to not recommended to have the user start the game in a room with a monster.
- Use a three dimensional (3-D) array in a way that adds significant value.
- Research and use a Java data container that supports multiple (heterogeneous) data types in a way that adds significant value.
- I am open to considering student suggestions.