**Second Individual Assignment**

**Due Sunday, March 15 by 11:59 pm**

Welcome to the second individual assignment in ITEC-3860! The goal from these series of individual assignments is to warm up your programming skills for the final project deliverable for your text-based adventure game. In games of this sort, the player wanders around from one location to another, picking up objects, and solving simple puzzles. The program you will create for this assignment is considerably less elaborate than the final project deliverable and it therefore limited in terms of number of rooms, items, monsters etc. Even so, you can still write a program that captures much of the spirit and flavor of the final game.

This handout contains what you need to know about the second individual deliverable along with a number of hints and strategic suggestions.

**Assumptions:**

1. Player is able to navigate between rooms
2. Map information (including rooms’ descriptions, connections) are being retrieved from text file
3. If you have not retrieve navigation command from text file in the first assignment I didn’t take points off but I will in the second assignment. So, make sure to get this feature implemented. Your navigation commands (N, E, S, W) should be retrieved from text file.
4. Your code is keeping track of visited rooms

**Goal:** implement the items and puzzle feature as directed below.

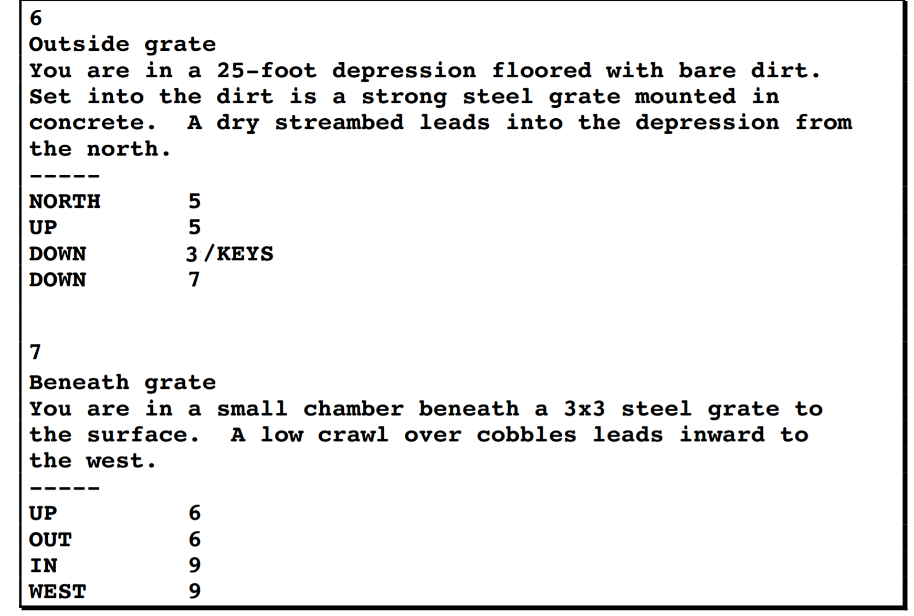
**Items feature:**

Now that the player is able to navigate between different rooms, for the second assignment deliverable your goal is to allow the player to interact with 3 different items of your choice in three different rooms. Interaction behaviuor with items should include the exact following commands:

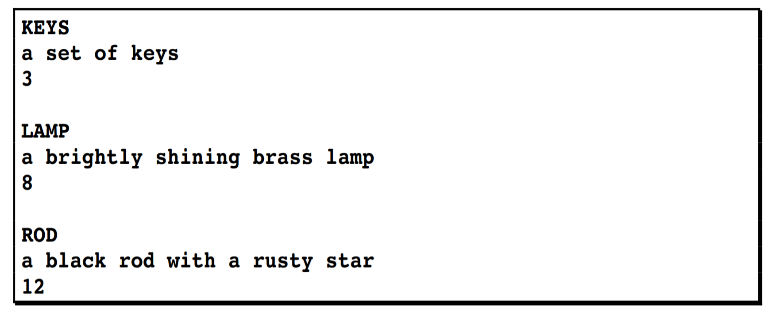
1. Examine item-name: this command will allow the player to retrieve the description of the examined item. Your game should display the description of the examined item to the console/GUI.
2. Pickup item-name: this command will add the item to the player inventory.
   1. Your game should display that the item has been added to the player inventory by displaying the following message to the console/GUI “Item-name has been picked up from the room and successfully added to the player inventory.
   2. Upon picking up an item from a room, the item should disappear from the room and the player should not see the item again when visiting the same room.
3. Drop item-name: this command should allow the player to access any item in the inventory and drop it in the current room.
   1. When an item is dropped, it should be dropped in the current room and be available for the player to interact with again for reexamine and re-pickup.
   2. Upon dropping an item, your game should display the following message to the console/GUI “Item-name has been dropped successfully from the player inventory and placed in room-name”

**Hint:** The items will move around in the game as the player picks them up or drops them off. Therefore, your implementation must therefore provide a facility (internal data structure) for storing objects in a room and in the player’s inventory of items. The easiest approach is to use an ArrayList, which makes it easy to add and remove items.

Under the assumption that you have followed the **suggested text file structure (You are allowed to use different structure as long as you are able to fulfill the requirements)** your text file for this deliverable could look like the one in the figure below:



You will need to consider adding another text file that holds the items information e.g. the figure below:



The entries in the items text file can consist of three lines indicating the word used to refer to the item, the description of the item that appears when you encounter it, and the room number in which the item is initially located. For example, this file indicates that the keys are in room 3, the lamp is room 8, and the rod is in room 12.

**Puzzle feature:**

Your goal is to allow the player to interact with one puzzle of your choice in any of the rooms. The interaction behavior with the puzzle should include the following commands:

1. Once the player enters a room with a puzzle, the game must display the puzzle description to the console/GUI and wait for the player to enter an answer.
2. Each puzzle has number of attempts allowed
3. If the player enters the correct answer, the game must display “you solved the puzzle correctly!” and the puzzle must disappear from the room. Note, the puzzle should disappear from the game and never show up again while navigating between rooms.
4. If the player enters wrong answer, the game will subtract 1 attempt from the allowed attempt and allow the player to provide an answer again. The game must display “the answer you provided is wrong, you still have number-of-attempt. Try one more time”
5. If the player is not able to solve the puzzle correctly after the given number of attempts, the game will display “failed to solve” message to the player and the puzzle must disappear from the game and never show up again while navigating between rooms.

Hint: in the text file, you can link the puzzle to the room similar to items. You need to consider adding puzzle text file to hold the following information puzzle name, description, answer, number of attempts etc.

**Important notes:**

1. Don’t send me screen shots of your code and don’t ask me to trouble shoot your code. This is not a programming class, but I found over the years of teaching this class that students need similar programming to review their programming skills and to avoid unnecessary drama toward the end of the semester. To help you I will discuss possible solutions in the class and show you sample code and you will put time and efforts to get your code up and running properly.
2. Use either JAVA Eclipse or IntelliJ
3. Grading code takes lots of time so keep your code clean, organized and understandable by adding comments. The more organized your code is the quicker I can grade and the faster you will get a feedback.
4. Don’t fix the file path in your code
5. If your code doesn’t run, you will get ‘0’ for this assignment. If your code runs but have partial behavior you will get partial credits. e.g. your code runs and allows me to navigate between rooms but doesn’t keep track of visited rooms, then I will take points off for this missing requirement.

The third assignment will add more to the first and second assignment. Therefore, it is important to finish this assignment on time and as directed. Otherwise, you will fall behind quickly.

**What you need to submit:**

1. Export your Java project folder, zip it and submit it on d2l. Don’t submit individual classes.
2. README file. The readme file should include explanation of each text file data structure. Clear map with room labels, connections, items and puzzles locations.
3. A clear screen shoot of given scenarios to show how to navigate, interact with items, and interact with puzzle.
4. Unlike the first assignment, this time you will get “0” if you fail to follow any of the above steps.