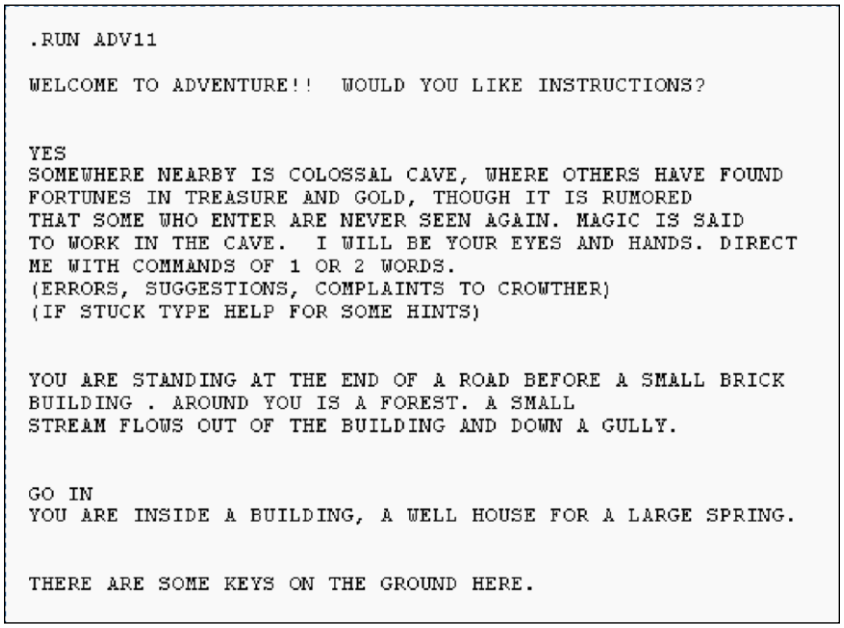
Moria Mines

(Textual adventure game)



```
.RUN ADV11

WELCOME TO ADVENTURE!!  WOULD YOU LIKE INSTRUCTIONS?


YES
SOMEWHERE NEARBY IS COLOSSAL CAVE, WHERE OTHERS HAVE FOUND
FORTUNES IN TREASURE AND GOLD, THOUGH IT IS RUMORED
THAT SOME WHO ENTER ARE NEVER SEEN AGAIN. MAGIC IS SAID
TO WORK IN THE CAVE.  I WILL BE YOUR EYES AND HANDS. DIRECT
ME WITH COMMANDS OF 1 OR 2 WORDS.
(ERRORS, SUGGESTIONS, COMPLAINTS TO CROWTHER)
(IF STUCK TYPE HELP FOR SOME HINTS)


YOU ARE STANDING AT THE END OF A ROAD BEFORE A SMALL BRICK
BUILDING . AROUND YOU IS A FOREST. A SMALL
STREAM FLOWS OUT OF THE BUILDING AND DOWN A GULLY.


GO IN
YOU ARE INSIDE A BUILDING, A WELL HOUSE FOR A LARGE SPRING.


THERE ARE SOME KEYS ON THE GROUND HERE.
```

Print terminal output of the world's first TAG: Will Crowther's original Colossal Cave Adventure aka ADVENT (1975-76), running on a PDP-10.

CONTENT

DESCRIPTION OF THE APPLICATION

Assignment

Analysis, design and implementation of a textual adventure game (TAG). A TAG is a dungeon described in text only. You will end up designing and implementing an application that lets you create a small, single player dungeon game. We have chosen to create this type of game because it lets you combine several parts of the curriculum encountered so far in a meaningful way, without having to introduce graphics handling, thus we can focus on programming.

The application must create a dungeon and a player who can walk through the dungeon by issuing text commands. The player must be able to fight monsters and pick up gold and items. The game ends when the player finds and picks up the "End Treasure Chest".

The project is divided into 3 milestones.

This is the description of Milestone 1, we will tell many more details about Milestone 2 and 3 later.

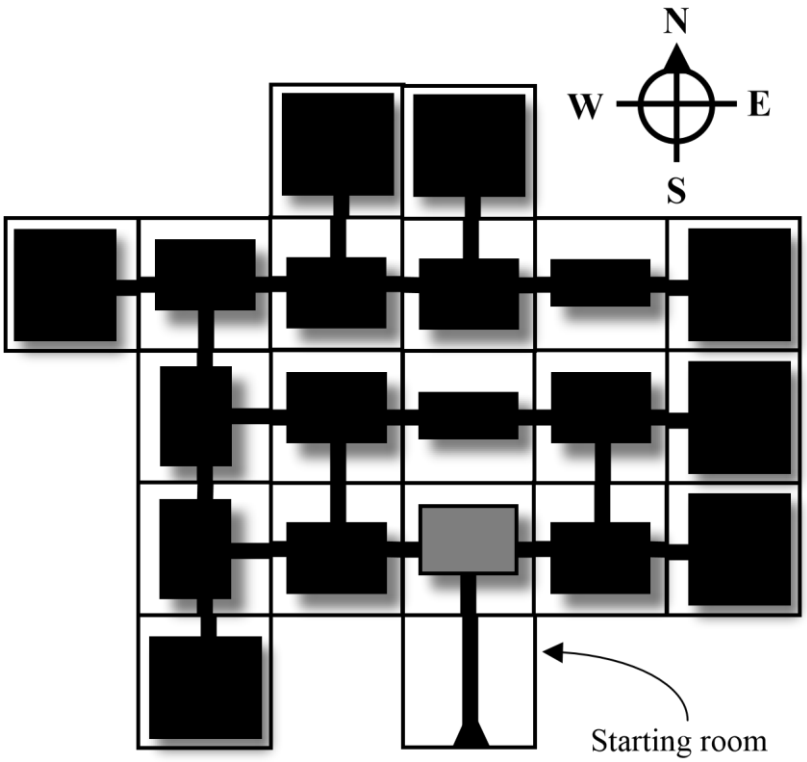
Milestone 1 purpose and requirements:

- To create a basic game with no persistence (The game starts over each time you play).
- The game must be able to read user input from the console and also output text to it
- You need to create a player that holds the amount of gold picked up so far
- You need to create a maze of rooms that the player can navigate
- Each room must have a text description and some gold that can be picked up.
- Furthermore, each room has up to 4 tunnels that leads to other rooms. The room "knows" what room each tunnel leads to.
- You must keep track of the rooms somehow.
- Otherwise, how you structure your code is up to you!
- Also, you are not forced to run a dungeons and fantasy theme, you decide the story and setting via room descriptions.

Enemies, combat, items, etc, will come in later milestones.

The dungeon

The TAG must implement a dungeon of several rooms. You may create your own dungeon or use the example dungeon drawn here



An example of a 20 room dungeon including 1 starting room. The light gray room has a south, east and west exit.

Each room must have a name, a description of how it looks and feels when standing in it. The room must know what exits lead out.

The exits/tunnels are always in the compass direction, so for instance the room after the starting room (the light gray room) has a west exit, east exit and south exit.

The game must NOT display a map of the complete dungeon or rooms where the player has been. Finding your own way around is one of the core game experiences.

Player

A player must have a name and health,

Starting

Starting the game:

Before the game starts, the dungeon must be created and populated with gold.
Secondly a new player must be created
The game must start in the starting room

Starting room:

There must be a starting room (displayed as a hallway/corridor on the map). This room must, in addition to the normal description, also contain an extra description with the adventure background and history: Why does the player enter this particular dungeon? Why is it so interesting and/or rewarding? Leaving the starting room by moving out of the dungeon must end the application.

Ending

You should make a special description in one of your rooms that tells the player he/she won the game by escaping from the dungeon and the amount of gold he/she got away with.

Action

The games must start the player in the starting room. For each room a description must be displayed. A player must be able to select an action in every room, for instance moving ‘north’ or ‘west’. The possible actions need not be immediately visible. It is part of the gaming experience that you must guess what actions can or should be taken in a room.

The actions must be text based, and use standard input. The game executes the action, for instance by moving the player north to the next room and showing him the description of this room. If a player does not write the correct command or writes a command that can not be executed, the game must respond appropriately and keep the game running.
At all times it should be possible to write ‘help’ and then be shown a list of **all** commands that the game accepts in its current state, and their exact spelling, even though only some of them is possible to execute in the players current room. This list should not show what commands are possible and impossible in the room you are in.

The following are commands that the player can select:

- **"gold"**: Shows how much gold is carried currently
- **"help"**: Shows all the commands that a player can type.
- **"quit"**: Ends the game. Display an appropriate end message.
- **"west", "east", "south" & "north"**: Moves the player to the room in the given direction. If it is not possible to move the player, the game must respond with an appropriate error message and let the player type a new command.

Play test

Test your dungeon and make sure it is possible to complete it.

HAND-IN AND SOLUTIONS REQUIREMENTS

Method

You will be working in groups of two on this project.

Design description:

1. Create a Class Diagram based on Developed Program. The diagram must show all elements used in the program. i.e.: all attributes, methods including parameters and return types and associations including multiplicity and navigation direction.

Programming:

Construct the Java application based on the specification in part 1 above. Specification of the hand-in

The assignment must be handed in on Moodle. The hand-in must include the following files:

1) The evaluation results in.*pdf* file format. (drag-n-drop into Netbeans project)
2) The Class diagram as a picture (drag-n-drop into Netbeans project)
3) The code exported by Netbeans to "zip" format.

The evaluation results
consists of:

- Results of the self-evaluation (See section named "checklist of evaluation criteria")

The code:

- Indentation should be consistent and follow a standard indentation style.
- Identifiers should be named so the program becomes easy to read

Evaluation / Learning objectives

The following criteria for the evaluation of the assignment are used by the teachers:

- The code does not break, and behaves correctly.
- The code is readable, i.e. indentation, naming of classes, methods and variables
- Data is encapsulated properly, i.e. proper use of visibility modifiers and getters/setters
- Code is maintainable and reusable, i.e. an object-oriented design where classes have a clear responsibility, i.e. controllers and entity classes

EXTRA INFORMATION

What is a TAG:

A textual adventure game is an old game genre that started in 1975 and still exists today. Many of the great computer game developers, like Richard Bartle, have started with TAG and moved on to develop other games. Denmark is famous for its DikuMUD from 1991 that favored hack-and-slash play style. It inspired other games.

Definition:

> "In online gaming, a MUD (multi-user dungeon), pronounced /mΔd/, is a multi-user real-time virtual world described entirely in text. (…)Traditional MUDs implement a fantasy world populated by fictional races and monsters, with players being able to choose from a number of classes in order to gain specific skills or powers. The object of this sort of game is to slay monsters, explore a fantasy world, complete quests, go on adventures, create a story by roleplaying, and advance the created character. Many MUDs were fashioned around the dice rolling rules of the Dungeons & Dragons series of games"
> (From http://en.wikipedia.org/wiki/MUD.)

If you want to try out a MUD yourself, then go here: http://www.topmudsites.com/

The first textual adventure game did not have much monster action, but was more of a puzzle game. You can find one implementation of this game on: http://www.amc.com/shows/halt-and-catchfire/colossal-cave-adventure/landing.

Voluntary extra options for ambitious students

- There could be a thief/tax collector in some rooms that take away some gold!
- Let the player have health, and add traps in some rooms that reduce health.
- Try to format the text descriptions nicely by using spaces and special characters. For example you can make a frame consisting of stars (like these: ********) around your room description.
- Make an end room with a special description and print out the amount of gold the player collected in his way.
- Some of you have maybe used arrays to arrange the rooms - Can you think of smarter ways than using an array to store the rooms?

Checklist for self-evaluation

Self-evaluation of Moria Mines (Textual adventure game)

Section A. Evaluation of the quality of the program code.

Please write in the box and/or ✓ or X against each of the points in second column below.

Evaluation of the quality of the code:

Criteria	Values	Tick (✓)
What are your test data? And does the program run on these test data?	We don't have any test data	×
Did you check that all your identifiers(variables, constants, classes and objects are appropriately	a. Name (e.g variables names are descriptive and should start with lowercase letters and class should start with uppercase etc.) b. Defined/declared (e.g check that variables(local, instance and static) are declared the right place, methods have parameters, return value type and body that reflects the method name) c. Initialized where appropriate value d. Invoked appropriately e. All identifies are used in your program to contribute to fulfill the program specification or have an appropriate role in the	✓

	program f. Access modifiers (private, projected and public) g. Scope and visibility of the identifiers understood	
Did you modularize your code so it is easy to understand?	a. Are there same pieces of code that are appropriate for method abstraction (redundant code) b. If a method is too long, it may be good idea to think about method modularity using method abstractions	✓
Control flow	a. All loop should terminate at some point in the program b. Switch statements should have a default case c. Avoid using multiple exit from a loop. Rethink about your algorithm if you think you need to do this d. Are there too many nested loops/conditions? Rethink about your algorithm if you think you need to do this.	✓
Input/Output	a. Does the program cater for all types of input? b. Are exceptions handles so that the program ends gracefully? c. Does the program run without breaking?	✓
Boolean expressions	a. Are Boolean expression is short and easy to understand with regard to the program logic?	✓
Documentation of the code	a. Is it clear from the comments that what the each segment of code will do? b. Do the codes do what the comments say for each appropriate segment?	✓

	c. Do the comments in the beginning of the methods explain what the method will actually perform? d. Do all the declarations(variable, class, methods) have appropriate comments? e. Are critical algorithms explained in plain language?	
Program layout	a. Indentation style is consistent. b. Code within a bloc (e.g. inside a loop) should be indented c. If a block is nested within another block the inner block's body should be indented relative to the enclosing block. d. Avoid excessive "stairstep" indentation. If problem reduce the number of spaces per indentation or switch to vertical style temporarily.	✓
Data encapsulation	a. Proper use of visibility modifiers and getters/setters b. Are local variables are visible only within the declared method, constructor, or block c. Access modifiers can be given for instance variables d. Instance variable are declared private e. Instance variables are decla in a class, but outside method, constructor or a block.	✓
Object oriented design	a. Does each class have distinct role e.g. controller class and entity class	✓

Section B. Evaluation against the program requirements.
Please write small note against each of the requirements below.

Requirements	**Your comments/notes**
Is you game to read user input from the console and also output text to it ?	Yes, our program gives output in the console, when the user uses the console
Does your show all elements used in the program. i.e.: all attributes, methods including parameters and return types and associations including multiplicity and navigation direction.	Yes
Did you create a player class that holds the amount of gold picked up so far	Yes
Did you create a maze of rooms that the player can navigate? Did you populate it?	Yes
Did you create a room class. Does each room object have a text description and some gold that can be picked up? Does the room have four tunnels? Which one is your starting room?	Floor class, that is same as a room class
How do you keep track of the rooms?	We have a coordinate system, and a map - handdrawn
How does your program end? What conditions makes it end of the game?	Death, or making your way to the end room where the treasure is.
Does the user get a menu of options to choose from once they enter a room? Do you have error-handling based on user input? Are there appropriate message for the wrong input? Do you have a mechanism for user to ask for help?	Yes, we have "invalid command" when the text doesn't match We also have a "help" command