Simple Text Adventure Game (STAG)

COMSM0086

Dr Simon Lock & Dr Sion Hannuna

Database Assignment

Marks & feedback for DB assignment on Blackboard
Feedback consists of three files:

- Test coverage
- Functionality
- Code Quality

Make sure you consider all feedback
When trying to interpret your grade

Overview

Final exercise is worth remaining 45% of assessment

The purpose of this assignment is to build...

<u>A general-purpose socket-server game-engine</u>
(for text adventure games)

If you are unfamiliar with the genre, take a look at:
https://tinyurl.com/zork-game

Game Server

The main class is a server listening on port 8888
Will accept incoming commands from clients

After processing a command from a user,
the server MUST close the connection
then listen for the next connection on port 8888
(in order to avoid congesting the socket)

Don't panic: this is provided for you in the template

Standard Gameplay Commands

- "inventory" (or "inv" for short): lists all of the artefacts currently being carried by the player
- "get": picks up a specified artefact from current location and adds it into player's inventory
- "drop": puts down an artefact from player's inventory and places it into the current location
- "goto": moves the player to a new location (if there is a path to that location)
- "look": describes entities in the current location and list paths to other locations

How can we play ANY game ?

Gameplay should be configurable by providing two "game description" files to the game engine:

- Entities: structural layout and relationships
- Actions: dynamic behaviours of the game

Before considering the content of these files
Let us discuss Entities and Actions at high level

Inheritance Hierarchy

Use a hierarchy of "Entity" classes in your game:

- Location: A room or place within the game
- Artefact: A physical "thing" within the game (that can be collected by the player)
- Furniture: A physical "thing", part of a location (that can NOT be collected by the player)
- Character: A creature/person involved in game
- Player: A special kind of character (the user !)

All entities will need a name and a description
Some sub-classes may need additional properties

The Location Class

"Location" is a complex entity which contains:

- Paths to other Locations (can be one-way !)
- Characters that are currently at a Location
- Artefacts that are currently present in a Location
- Furniture that belongs in a Location

Actions

Dynamic behaviour within game is represented by "Actions", each of which has following elements:

- A set of possible "trigger" key phrases (ANY of which can be used to initiate an action)
- A set of "subject" entities that must be available (ALL of which be present to perform the action)
- A set of "consumed" entities that are removed (ALL of which are "eaten up" by the action)
- A set of "produced" entities that are created (ALL of which are "generated" by the action)

Example Actions

The previous description of Actions
may be a little hard to comprehend !

Let's take a look at some examples to illustrate...

actions

Document Formats

<u>DOT</u>

A language for expressing graphs

(which is basically what a text adventure game is !)

entities

<u>XML</u>

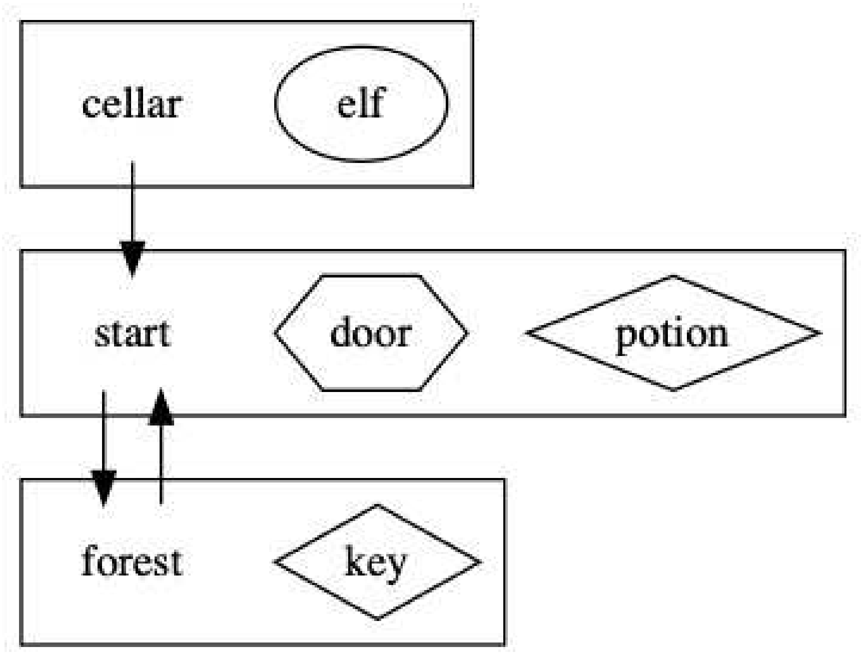
A language for representing structured data

actions

Visualising the DOT files

The big bonus of using DOT files is that there are tools for visualising them (GraphViz !)

We can SEE the structure of the "entities" file:



Online Editor

20%20%20%20%20%20tree%20%5Bdescription%20%3D

Parsers

You've already experienced writing parsers
We don't want to cover old ground
So you'll be using two existing parsing libraries !

There is considerable educational value in
learning to use existing libraries and frameworks

This can get lost in a desire to learn fundamentals
But not on this unit !

Which parsers ?

For parsing DOT you should use "JPGD":

http://www.alexander-merz.com/graphviz/doc.html

Library should already be embedded in maven project

And the Java API for XML Processing "JAXP":

http://oracle.com/java/technologies/jaxp-introduction.html

Core library which should be available to your project

Specifying Game Files

The two game configuration files should be passed as parameters into the GameServer constructor:

```
GameServer(File entitiesFile, File actionsFile)
```

This is essential so different games can be played
We have a special "marking" game configuration
So make sure your code can read in config files !

Unique Identifiers

Entity names in the configuration files are unique
So you can safely use these as unique identifiers

You won't have to deal with two things called "door"
There might be a "trapdoor" and a "frontdoor"

Think of them as variable names !

Action phrases are however NOT unique
(it might be possible to "open" many things)

Command Interpreter Flexibility

You interpreter needs to be able to cope with:

- Varying Case: All command are case insensitive
- Decorated Commands: extra "unnecessary" words
- Varying Word Order: triggers/subjects in any order
- Partial Commands: some subjects not mentioned

Full details for all of these contained in workbook

Correctness and Certainty

Your server should block any commands that contain extraneous entities

Stated by user, but not defined as subject of action

`open door with key and axe`

Your server should perform no action if command is ambiguous

Command matches more than one action

`open with key`

(if there are two doors, each with an "open" action)

Player Identification

Incomings command begin with a username
(to identify which player it is)

A typical incoming message might take the form of:

```
simon: open door with key
```

This allows the game to support multiple players !

Server does not need to deal with authentication

Health

As an extension to the basic game...
Add a "health level" feature to the Player class

Each player should start with a health level of 3
Poisons & Potions increase and decrease this number
(See the health related actions in the "Actions" file)

When player's health runs out, they return to start

Add a new "health" command keyword...
That reports back the player's current health level

Testing

A special set of custom game description files will be used to assess your game.
It is therefore essential your code is able to parse files in the same format as examples provided !

Scripts will be used to automatically test your game engine to make sure it is operating correctly.
It is therefore essential that you adhere to the "built-in" commands detailed previously !

Code Quality

Code quality will again be used during marking

Adhere to guidelines outlined in the workbooks

Also review the quality feedback advice from DB !

(try to improve your bad habits !)

Plagiarism

Automated checkers used to flag possible plagiarism
If markers feel plagiarism may have taken place...
The incident is referred to faculty plagiarism panel

May result in a mark of zero for assignment
or even the entire unit (if it is a repeat offence)

6 students were referred for the DB assignment
They will be contacted shortly by the school office

Deadline - Thursday 4th of May @ 13:00

Questions ?

Demo of Server in Action

RunGameServer

In order to connect to the server
We have also provided you with a GameClient:

RunGameClient

You won't need to alter the client code !