Object Oriented Programming with Java

Introduction to the Next Assignment

Dr Simon Lock & Dr Sion Hannuna

Simple Text Adventure Game (STAG)

Overview

Final exercise is worth remaining 60% of assessment

The aim of this assignment is to build...

A general-purpose socket-server game-engine
(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
This accepts incoming commands from clients
It then processes command & changes game state
Returning a suitable response back to the client

After processing command server closes connection
Then listens for the next connection on port 8888
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 in the possession of the player
- "get": picks up a specified artefact from current location and adds it to the 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 (only if there is a valid path to that location)
- "look": describes the current location, including all entities in that location and paths to other locations

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 !
It is really just there so you can play the game

With this Server, you can play ANY game !

How is this possible ?

Gameplay is configured by providing two separate 'game description' files to the game engine:

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

Before considering the content of each of these files
Let's discuss 'Entities' and 'Actions' at high level...

Different Types of Entity

- Character: A creature/person involved in game
- Player: A special kind of character (the user !)
- 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 CANNOT be collected by the player)

The Location Class

Locations are a complex entity which can contain:

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

Actions

Dynamic behaviours are represented as 'Actions'
Each 'Action' may have the 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'
is probably a little hard to comprehend !

Let's look at some examples by way of illustration
They are represented using a language called XML
(the same language used by the Maven POM file !)

actions.xml

Entity Files

Rather than also representing entities using XML
We will use an alternative language called 'DOT'
(It's good to experience a range of languages)

DOT is a language used for representing graphs
(which is basically what a text adventure game is !)

entities.dot

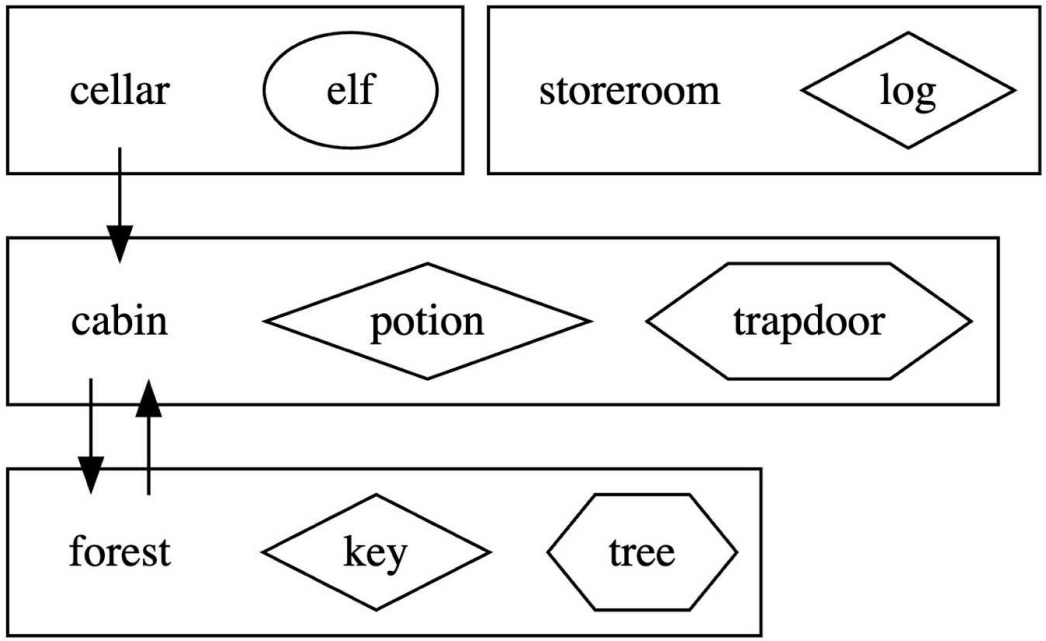
Sorry about extension name - it's not my choice !

Visualising the DOT files

The BIG bonus of using DOT files is that...

there are many tools for rendering them (GraphViz !)

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



Online Editor and Visualiser

https://dreampuf.github.io/GraphvizOnline/

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

Parsers

You've already gained experience writing parsers
We don't really want to go over "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 files, 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 that _should_ be available to your project

Command Interpreter Flexibility

Your 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 trapdoor with key and axe
```

Server must NOT perform an action if the command is ambiguous

Command matches more than one possible action

```
open with key
```

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

Player Identification

Incoming commands always begin with a username (to identify which player has issued that command)

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 (that would add to the complexity of assignment)

Annual Variation

Every year we try to vary the assignments a little
Just to make it a bit different from previous years

We have already used most functional variability
So we now start to explore code-level variations

This year we introduce a set of "illegal" constructs
These are features of Java that YOU MUST NOT USE

Illegal Constructs

- Lambda operator "->" (there's always an alternative)
- Arrays and ArrayLists (use more "exotic" structures)
- Ternary operators (aka "the Excel IF statement")

```
String partOfDay = (hour >= 12) ? "PM" : "AM";
```

- Unqualified method calls (you MUST use "this")

```
this.checkForWin()  rather than  checkForWin()
```

- + operator on strings (use printf and StringBuilder)

Such constraints may seem annoying and illogical...
But rules are rules !

That's a bit Strange ?

To help ensure you're not using "illegal" constructs
We've provided a sourcecode checker for you to use

This checker is called "strange" (long story)
Run it from the command line using maven exec:

```
mvnw exec:java@strange -Dexec.args=<source-file>
```

NOTE: strange only checks 1 <source-file> at a time
So you need to run it for EACH file that you write:

```
...-Dexec.args=src/main/java/edu/uob/GameServer.java
```

demo

Word of Warning

"Strange" is very much a BETA version
It may produce some "unexpected" results

If it flags an issue with your code, do try to fix it
If you think "Strange" is wrong, let us know

During marking, we will manually check any issues
It won't take us long to look through the reports
(since there shouldn't actually be any illegal code)

Testing

A special set of custom game description files will be used to assess your game during marking. It is therefore essential your code is able to load 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 assessed during marking

Adhere to guidelines outlined in the quality lecture

Derived Code

We'll use analysis tools to determine "derived" code

Material that was "found" online or generated by AI

Any derived code will be discounted during marking

You'll only receive credit for code YOU have written

Collusion

This is an individual assignment, not group activity
Automated checkers used to flag possible collusion
If markers feel collusion has indeed taken place...
Incident is referred to academic malpractice panel

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