SENG201 2014 Assignment Report

Andrew Dallow - ID: 56999204

Introduction

The task of the assignment was to develop a simple world simulation of places, things and actors. Places have a label, description and two-dimensional size (width and depth) as well as the ability to contain things, such as, furniture, equipment and food with a name and description. Actors have a name and are able move between places, taking and dropping things a place's contents as they go. A reporting system was also needed in order to track who actors are, where they are and what they are carrying as well as giving the details of each place and what it contains.

Implementation

Modelling things in the world required a Java class called 'Thing' with the ability to set and get a name and description of the Thing. For displaying basic details in the reporting system a Java toString method was implemented.

'Plant', 'Food' and 'Computer' classes extend more specific cases of 'Thing'. 'Plant' has properties of plant type (e.g. tree, etc.) and quantity because there are many different types of plant which commonly exist in large numbers, such as in a forest. 'Plant' has a custom toString method specifying quantity along with the name and description.

The 'Food' class has the ability to specify food type (e.g. fruit), since there are many food type, and whether it is cooked or not. 'Food' has a custom toString method which specifies if it is cooked or not along with its name and description.

'Computer' models the simple behaviour of computers. Computers are usually identified by a number, therefore an integer number is used for the computer name along with the description in 'Thing'. Computers can also be logged onto by a person in the same location and used to control a robot. 'Computer' implements a toString method to display who is logged on and which robot it may be controlling in addition to name and description.

The abstract class 'Place' models places with the properties label, description and the size (width and depth in metres). 'Place' contains a collection of type 'Thing' giving it the ability to store objects of type 'Thing', representing the contents of the place. Things can be added to the collection via the 'addThing' method and a comma-delimited string of all Things in contents is obtained via the 'contentsList' method. The toString method is an abstract method, therefore classes extending 'Place' must implement their own version of toString.

'Room' and 'Forest' extend more specific cases of 'Place'. 'Room' models rooms in the world, has the additional property of level, specifying the story the room is located on. 'Room' implements a toString method containing room label, level and description.

'Forest' models forests in the world, having the properties forest type, climate, and plant density, since there are many different types of forest (e.g. Alpine, etc.) with varying densities of plant life and various climates (e.g. warm, cold, etc.). Dimensions of the forest and its plant contents are used to calculate forest density (plants per metre squared). Custom methods, 'addPlants' and 'removePlants' are used whenever plants are added or removed, calculating the forest density. 'Forest' implements toString, displaying forest density in addition to the label and description.

The abstract class 'Actor' has properties of actor name and current location. The location is initialised to null and then one can use the moveTo method to change the location, essentially moving the actor between places. Location history is stored in a collection of type 'Place', updated when an actor moves between places, giving the actor a 'memory' of past locations. Actors have an inventory as a collection of type 'Thing' allowing them to store things on themselves. The 'Take' and 'Drop' methods are used to add and remove items from the inventory in the current location. The toString method is an abstract method, therefore classes extending 'Actor' must implement their own version of toString.

'Person', 'Robot' and 'Animal' extend more specific cases of 'Actor'. 'Person', which models people in the world, has the properties of date of birth (DOB) and gender. DOB is used to calculate the age of the person which is used in toString along with name, gender and location. 'Person' also has an 'eat' method for Things of type 'Food' in their inventory, removing the Food from the world.

'Robot' models robots in the world, having the additional properties of purpose, computer controller and a powered on/off status. A 'Robot' has to have an integer identification and a purpose (e.g. lift heavy items) and in order use the 'moveTo', 'Take' and 'Drop' methods it must be powered on, a 'Computer' has to be set to control the robot and a 'Person' must be logged on to that computer. The toString method displays the robots ID, purpose and current location and if it is powered on or off.

'Animal' models animals in the world, having properties of type (e.g. bird, mammal), number of legs and gender. An 'Animal' can take and drop items like any actor but can only move between places of type 'Forest'. 'Animal' also has an 'eat' method similar to

‘Person’ but can consume food in the contents of its current location. The toString method displays the animals name, type, number of legs and current location.

‘World’ models entire worlds which has the ability to store a collection of Actors, Places and Things. Using collections in this manner allows the world to easily modify and keep track of objects created in the world. ‘World’ also has a demoWorld method which creates a demo world to demonstrate the capabilities of the program.

The ‘Reportable’ interface has the methods ‘report’, setDestination(PrintStream) and setDestination(PrintWriter) so that any class implementing the interface can produce a more detailed and appropriately formatted report than is practicable with toString. The interface was used with the class ‘Reporter’ to confirm that the model conforms to the expected behaviour.

To keep with the Model-View-Control concept, two classes are used for the graphical user interface (GUI). ‘WorldView’ views changes in the model by implementing the ‘Observable’ interface and making the ‘World’ observable. ‘World’ also implements ‘Observable’ in order to observe changes in Actors, Places and Things, thus propagating the observed changes to ‘WorldView’. Methods in the model which change the state of an object have all been made observable (e.g. setters, ‘moveTo’, ‘Take’ and ‘Drop’ etc.). ‘WorldView’ also displays a text GUI that prints out messages representing changes in the world. For example, if a person moves between places the message “Update involving: Andrew Dallow moving to 112.” will show up in the GUI. This informs the user that a change has occurred in the model.

‘WorldControl’ contains the GUI for controlling all aspects of the model. When a change occurs in the model, ‘WorldView’ notifies ‘WorldControl’ and updates the GUI accordingly. The ‘WorldControl’ GUI consists of four main panels: Places, Place Contents, Actors and Actor History and Inventory.

The Places panel contains a selectable list of Places in the world, where the toString methods of ‘Room’ and ‘Forest’ are used to give the details of each Place. Below the list are three sub-panels with text boxes and buttons which allow the addition and editing of Places, as well as setting their dimensions. The Place Contents panel contains a selectable list of Things in the selected places contents, where details of the Things are displayed using the toString methods in Things. Below this list are four sub-panels with text boxes and buttons for adding new Things, Foods, Computers and Plants into the world and the contents of the selected place. The Computer pane contains an extra button for setting a selected computer to control a selected Robot.

The Actors pane contains a selectable list of Actors, where the toString methods of actor classes have been used to display the details of each actor. Below this list are three sub-panels with text boxes and buttons for adding new Persons, Robots and Animals to the world. Below the Actors pane is the Actor’s History and Inventory pane. The history pane lists all past location of the selected actor. The inventory pane is a selectable list of Things in the inventory of the selected Actor. In both cases toString of ‘Place’ and Thing classes are used to display the details of the respective objects.

The bottom of the GUI contains ten buttons which are used for changing the state of an object in the world. The ‘Go To’ button changes the selected Actor’s location to the selected Place. The ‘Take’ button moves a selected Thing from the contents of a Place to the inventory of a selected Actor in the same location. The ‘Drop’ button is similar but instead the Thing is removed from the selected Actor’s inventory and added to the Place’s contents. The ‘Log On/Off’ button logs the selected Actor on or off a selected computer in the contents of the Actors current location. The ‘Eat Food’ button is used by either a selected Person to eat a Food item in their inventory or for a selected Animal to eat food from the contents of their current location.

The ‘Demo World’ and ‘New World’ button, when clicked, create a new demo world with pre-defined settings or a blank world, respectively. The ‘Remove’ button is used to remove any selected object from the world. The ‘Clear’ button clears all selections over all lists, while the ‘Quit’ button ends the program and closes the windows.

Incorrect user input has been considered in all features which affect the state of the model. That is, if a user performs an incorrect action, for example, leaves a text box blank, enters the wrong text format, or selects incompatible objects, then a JOption dialog box pops up informing the user of the error.

Conclusion

Overall, the world simulation program simulates places, actors and things in the world with some extended, more specific cases. Places can have a name, description and contain a collection of Things. Actors also have a name and the abilities to move between places taking and dropping Things as they go. Things have a name, description and some types have the ability to be interactive with actors, for example, computers and Robot. A GUI reporting system was implemented in order to display and track the current state of the model as well as the feature for adding new Places, Actors and Things to the world. Therefore, this program meets the specifications of the project.