COSC 3P91 – Assignment 1 – 6920201 & 6973523

BRETT TERPSTRA & MICHAEL BOULOS, Brock University, Canada

1 Packages

There are a total of 4 packages in the project, which are appropriately represented within the UML class diagram, ‘game’, ’gameobjects’, ‘userinterface’, ‘player’.

Package	Description
game	This package contains classes such as Map and Engine which determine background game behaviour such as village generation and represent the underlying map from which all other classes interact with – including the player.
gameobjects	This package contains classes such as Building & Inhabitants, and their derived subclasses and helper classes, that represent the units and village elements that need to be built or produced.
userinterface	This package contains the graphical user interface manager, it interacts with the game to produce a GUI that the player will use to play the game.
player	This package contains the Player object, which appropriately represents the player system. This will be modified to support multiple players later on.

There are a variety of classes in the game source; each one plays a unique role in facilitating the producing, building, upgrading, exploring, attacking game loop. Below are some the classes and explanations of their attributes and operations that better demonstrate how the game will function.

2 Building Classes

A basic building object extends from the abstract class `Building` and represents a built structure in a village that can be upgraded, among its own special operations. A building has an upgrade level, health, cost, and build time (the time it takes from starting construction to finishing the building). A building is placed on a map and takes up a certain number of tiles, a tile is represented by an instance of the `Tile` class.

2.1 Village_Hall *extends* `Building`

The village hall is the heart of a village, represented by the class Village_Hall – it has attributes that manage the wood, iron, gold capacity of a village and their respective getter methods. A village will only have a single village hall, hence the inclusion of a single

‘townHall’ attribute within the Map class. Stages within a village hall determine the resource capacity a village has, and hence how much of each resource can be harvested and stored by a village; the higher the stage of a village hall, the more resources a village can store.

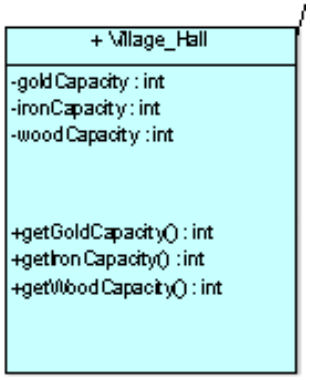

Fig. 1. The Village_Hall class as produced in the UML class diagram

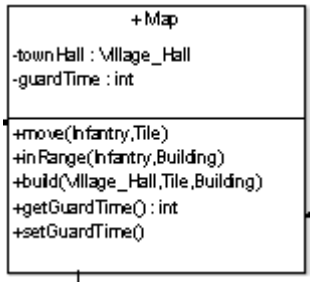

Fig. 2. The Map class as produced in the UML class diagram, notice the townHall attribute which specificies the village’s Village_Hall object

2.2 ResourceBuilding *extends* Building

A resource building is a building unit in a village that is responsible for producing resources (gold, iron, & wood) for the village, which are held by the Player object -- since not all displayed maps will belong to the current player but a player can access the resources of their own village. Any class that inherits from resource building will carry its own static resource name, to specify what its responsible for harvesting depending on its type, a harvest rate to specify how fast it can harvest, and a harvest operation to contribute a resource to the village hall – a specific village.

The buildings that inherit from this class are the iron mine, lumber mine, gold mine, and farm, however, the farm also adds a method to return its population contribution since farms contribute to the number of inhabitants in a village. All resource buildings also have a generalization of the stage object, a ResourceStage, which adds the stage attribute of a harvest rate increase for their harvest functionality.

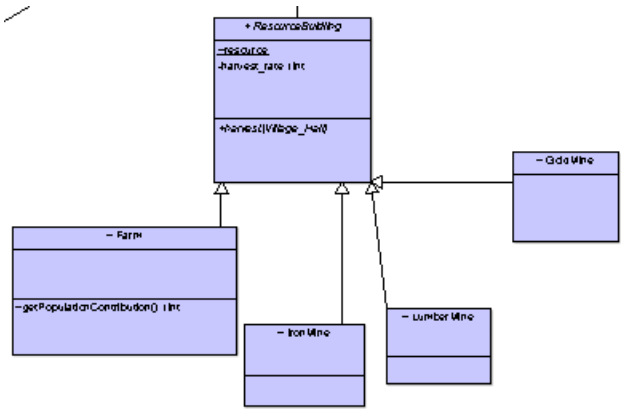

Fig. 3. The ResourceBuilding class as produced in the UML class diagram and its subclasses

2.3 DefenseBuilding *extends* `Building`

A defense building is a building unit in a village that acts as a line of defense from attacking infantry. In addition to the inherited members from the building class, a defense building also has damage and range attributes that correspond to the damage and range of its attacks; a defense building comes with an attack method, targeted towards an instance of an attacking infantry unit specified by the parameter, to facilitate the functionality of defense.

The buildings that inherit from this class are the cannon and the archer tower. All defense buildings also have a generalization of the stage object, a `DefenseStage`, which adds the stage attributes of the change in damage and change in range tied to the new stage.

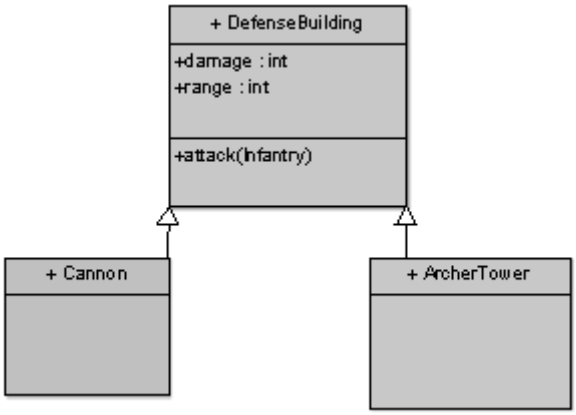

Fig. 4. The DefenseBuilding class as produced in the UML class diagram and its subclasses

3 Inhabitants and Tiles

3.1 Inhabitants

Classes that implement the `Inhabitant` interface represent the inhabitants that will populate a village, those who may 'work' and 'build' within the villages or may act as infantry when attacking other villages. Using its move method, an inhabitant may move to coordinates on a village map (represented by a `Tile`), this affects their current position which can be retrieved by a position getter method. The amount of the inhabitants will be determined by the amount of farms in the village, and how much food is being contributed to the population.

The classes that implement the `Inhabitant` interface are the worker, collector, and infantry classes. The workers contribute to the production of food and the construction of buildings by interacting with the village itself, the collectors interact with one of the respective resource building instances to facilitate the harvesting of resources, and the infantry class splits up into designated subclasses, specializations which determine their attributes and affect the amount of health, damage, and range they have.

Infantry classes are more complex in their functionality as they are bound to the attacking functionality of the game, thus maintaining health, damage, and range properties and utilizing an attack method to attack nearby target buildings.

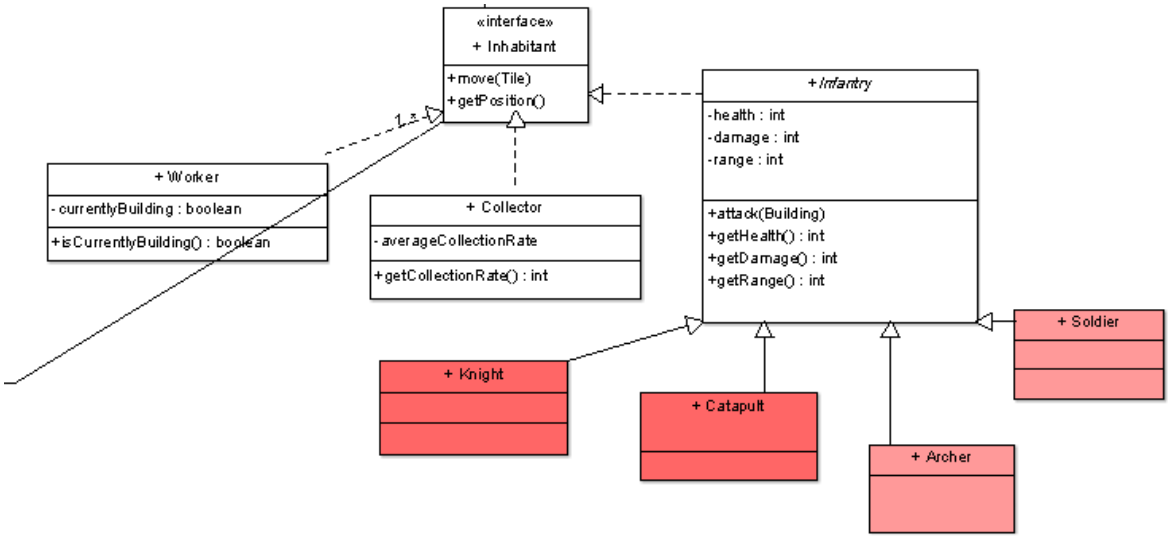

Fig. 5. The Inhabitant interface as produced in the UML class diagram and the classes that implement it

3.2 Tiles

A tile represents a unit of space within a village with its own map coordinates; tiles associate closely with buildings as they are the spaces on which a building is built on. Many game objects utilize tiles to specify coordinates in a village, commonly during interactions with the map.

As a design note, tiles associated with units will be used later on to determine how to draw certain GUI elements.

3.3 GameEngine and Map

The game engine and map are unique classes that are crucial to facilitating the game loop. Their methods and members are fairly transparent regarding what role they play in aiding the class to fulfil this purpose.

DOCUMENT END