CSE 1325-001
Course Project Phase 2
Archaeology
Assigned: October 1, 2014
Due: November 3, 2014

1. Instructions

Students are to continue the assignment by updating ***a pair*** of Java programs. These programs need to be complete and able to run in order to receive full credit. All work, diagrams, programs, and submitted outputs, should include the student name, UTA id number, and the date. ***You will need to demonstrate your code to Dr. Becker, Mr. Robinson, or other Teaching Assistant.***

The project may be done in teams of two. In order to do so, teams must remain the same from Phase 1 of the project.

To turn in the Java code for this assignment, create a zip file of the working directory, and rename the file Phase2.TeamName.zip, where the TeamName is, of course, the name of your team. Then, upload this assignment to Blackboard. In addition, you must upload copies of any written documentation or diagrams to Blackboard.

2. Objective

The goal of this project is to cover the introductory material of the Java language, including

- Class Properties
- Inheritance
- Interfaces
- Polymorphism
- Static Members
- Exception Handling
- Beginning Graphical User Interface (GUI)
- Menus and Menu Items
- Basic Dialog
- Text Fields and Scrollbars
- The Subcontroller Concept
- UML Class Diagram

3. Problem

Update two programs: one will be the map creator program (Map Population Tool), and the other will be a map reader program (Archaeological Dig Tool). These programs are to be built in an Object-Oriented manner. In addition to the programs, an archaeological map will have to be created. The map must be at least 50% full of archaeological features and finds. Maps, features and finds are defined as follows. In addition, two **UML Class diagrams** must be created, one for each tool.

4. Diagram and output.

Create a ***class diagram*** of each program, showing how the classes relate to each other. The diagram must have a written description, preferably a dictionary of terms of the parts of the diagram. These diagrams must be generated showing the aggregation, association, and generalization (inheritance) arrows. There should be two diagrams: One for the map maker, and one for map viewer. Turn in a screen capture of a finished archaeology map from the ADT tool to Blackboard. This screen capture must show a 16x16 grid, and populated with 50% (128) of features.

5. Design Updates

The programs will have two key updates: new properties from inheritance and beginning GUI with a text area.

5.1. New Properties

5.1.1. Demonstrate Inheritance

For this update, inheritance will also need to be shown. In particular, the inheritance is to be added to the “Finds”, that is the pottery, the charcoal, and the metal work.

- Pottery will be able to have three subclasses: Storage, Decorated, or Submerged.
  - Storage Pottery has a double field that indicates the volume of the vessel.
  - Decorated pottery has a String data member that will contain one word to describe the pottery. Example: “Sacrifice”,”Moon”,”Sun”.
  - Submerged pottery has an integer field to represent depth in meters. In many cultures, pottery and other artifacts are an offering in water-filled environments (Fens, Cenotes, etc)
- Charcoal deposits have two subclasses, either a kiln for making goods, or a hearth for cooking food.
  - Kilns are a circular feature, will have a data member that is the radius in meters.
  - Hearths are a rectangular feature, and will have two data members to describe its length and width in meters.
- Metalwork deposits will have two subclasses, either ferrous (iron based) or non-ferrous.
  - Ferrous objects will have an integer signal strength indicator, the stronger the indicator, the larger the iron object.
  - Non-ferrous objects will have a String data field to indicate what the different metals are: Copper, Bronze, or Gold.

As a result of all these combinations, a text report of what is found, how much, their dates, and their properties have to be displayed to screen. This will be discussed in the View Menu below.

5.1.2. Demonstrate Polymorphism

One of the more interesting aspects for Object Oriented programing is Polymorphism. First, you are to use Polymorphism to allow the collections of finds for your MapData to hold the children of the various finds. For example, the ArrayList of pottery objects would be able to contain classical, decorated, and submerged objects. Do this for the charcoal and metalwork object lists as well.

Second, make the metalwork parent class an abstract classes, because each of them will have an abstract method that must be overridden. For the Metalwork Find, there will be an abstract method called respondToMetalDetector(), which will return a specific integer. If at least one find is ferrous, return a value of 2. If at least one find is non-ferrous, return a value of 4. If there are no metal finds, return a value of 0. If you should have a map grid square that has both ferrous and non-ferrous components, return a 6.

5.1.3. Demonstrate Static Behavior

In the Map Population Tool (MPT), only one gold item can ever be placed on the map. Use static properties to make sure that only one gold item ever exists anywhere on the map. If a second gold object occurs, the first should be either removed or overwritten. Several solutions exist, be sure to be able to demonstrate yours.

5.1.4. Demonstrate Custom Exception Handling

Add a new function to your map to assign a new property to all map squares. Each square can now be marked "heritage". This means archaeologies may not dig into this grid square. If the user tries to dig into this square, the function should throw a custom exception that is handled within the program.

5.2. GUI Components

Next, let us consider Graphical User Interfaces (GUI). For this phase, you are to build a few key components. The GUI would consist of a Frame that runs the main program, and upon the frame would be menus. The menus would replace the text menus from phase 1. Each menu item clicked would open an input dialog box. The input dialog box would be a series of simple dialog boxes that take the required information. (Row, column, feature, find, etc)or execute a particular command (Load, save, exit). The bulk of the frame would contain a text area where the output of the program and the maps will be displayed.

5.2.1. Breaking apart GUI and Process: The Subcontroller Concept

Various IDEs and other projects will actually generate code for the programmer, which is a marvelous time-saving function. Only one problem: Connecting to the rest of the process and data management. Never put any of your key function code inside the GUI itself. Logic and mathematics should be at a level below. Be sure to include in your design a class called Subcontroller. All the commands from the GUI will be routed through the subcontroller before reaching the rest of your code. This way, when the GUI changes or is updated, the functionality that handles and process data would remain untouched.

5.2.2. The Text Area

The main body of the applications will be a text area that will act as the console for the programs. Instead of System.out.println to screen, the current output will be displayed in this *text area*.

5.2.3. The Menus

The frame will contain a MenuBar, which will have four menus and a menu item. They are File Menu, Edit Menu, View Menu, and About Menu Item.

5.2.3.1. File Menu

On both programs, the File Menu should have three Menu Items: Load, Save, and Exit. You may either use a dialog to prompt for the filename, or you may hardcode the filename in Phase 2. (Phase 3 is a different matter).

5.2.3.2. Edit Menu

5.2.3.2.1. Map Population Tool

The Map Population Tool has to cover the functional requirements from the first version of the program, and the new functions as well.

- Create Map – New Map of x rows by y columns
- Update a map grid square Feature: Add a single Stone, Pit, or Natural
- Update a row of map grid squares: Add a whole row Features
- Add a Find: Pottery, Charcoal, Metalwork or their children to a map grid square.
- Mark a map grid square "heritage"

5.2.3.2.2. Archaeological Dig Tool

The Archaeological Dig Tool has to cover the functional requirements from the first version of the program, and the new functions as well.

- Scan a map grid square with a Magnetometer
- Scan a map grid square with a Metal Detector
- Dig a map grid square with a Trowel

5.2.3.3. View Menu Item

The two program tools will have to be able to show the current maps to the user, so the user editing the maps has an idea of what they are seeing. Also,   Both tools will have to be able to update the character symbols of their maps.

The Map Population View Menu

- Update Map Symbol – Choose the symbol for the type of terrain to be displayed
- View Current Symbol Map – Show the current x by y map of rows and columns of features.

The Archaeological Dig Tool View Menu

- Update Map Symbol – Choose the symbol for the type of terrain to be displayed
- View Current Symbol Map – Show the current x by y map of rows and columns of features.
- View Magnetometer Map – Show the current x by y map of True/False for magnetometer
- View Metal Detector Map - Show the current x by y map of True/False for metal detector
- View Pottery Finds Map – Show the current count of pottery finds
- View Charcoal Finds Map – Show the current count of charcoal finds
- View the Metalwork Map – Show the current count of Metalwork Finds
- View Report – At this point, a list of all finds (metal, pottery, charcoal, and their children) with their dates and properties will be listed. The first table will be pottery finds of all types, sorted by their date. The second table will be charcoal finds of all types, sorted by their date. The third table will be metalwork finds of all types, sorted by their date. Each row of the table will also include the row and column the find was found, and their special properties from section 5.5.1. At the end of this list of finds, the average date of the site will be displayed.

5.2.3.4. About Menu Item

On Both Programs, the Menu Bar will have a menu Item called About. When About is clicked, a dialog box should open that shows the following information: The Team Name, The first Team Member Name and ID Number, the second Team Member Name and ID Number, the name of the tool (Map Population Tool or Archaeological Dig Tool), and the date the program was finished. The about box should also include "Version 0.2".

CODING STANDARD

Writers of Java code have established some simple standards for how a program should be written, addressing both the functionality and the readability of the code. You can find several standards online by simply searching for Java coding standards, including ones by Oracle and Google. Additionally many corporations who use Java have established their own, slightly modified, set of coding standards.

To help you get used to coding to a standard Phase 2 of the project will include a chance to get 5 points of extra credit on the ***midterm*** by adhering to the following standards in your code:

1. Variable names will be written in camel case (e.g. mapData or mainMapMenu). This includes both the member variables of your classes and the internal variables used by a method, such as the parameters.

2. Class names will be written with the first letter of each word capitalized (e.g. MainMenu, MapNode).

3. Class and variable names will be in plain language and will not be abbreviated (e.g. instead of calling an instance of MainMenu simply mm you would call it mainMenu).

4. Any member variables of a class will be set as "private" variables. In order to access those variables getter and setter methods will be created.
e.g.

```
public class MapData
{
	private MapNode mapNode;

	public MapNode getMapNode()
	{
		return mapNode;
	}
	public MapNode setMapNode(MapNode newMapNode)
	{
		this.mapNode = newMapNode;
	}
}
```

5. JavaDoc style comments are included for all classes and methods as in the previous assignment.