CSE 1325-001
Course Project Phase 1
Archaeology
Assigned: September 8, 2014
Due: September 29, 2014

1. Instructions

Students are to complete the following assignment by writing ***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 together throughout all phases of the project. Students wishing to work as a team must submit their request to the instructor in an e-mail including both names and both UTA student ID numbers and a Team Name.

To turn in the Java code for this assignment, create a zip file of the working directory, and rename the file Phase1.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

- Use of basic data types
- Controls
- Simple concrete classes
- Files
- User Input
- Screen Output
- Exception handling
- Collection types

3. Problem

Create two programs: one will be a 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 top-level abstract diagrams must be created for each tool.

4. Diagram and output.

Create a ***top level abstract diagram*** of each program, showing how the various programs and their components relate to each other. The diagram must have a written description, preferably a dictionary of terms of the parts of the diagram. There should be two diagrams: One for the map maker, and one for map viewer. Also turn in a printout of a finished archaeology map to Blackboard. This test map must be 16x16, and populated with 50% (128) of features.

5. Map Populating Tool

The Map Populating Tool (MPT) will be a menu-driven program that has the ability to create a data-file that can be processed by the second program, the Archaeological Dig Tool (ADT where tools can be applied to the viewable map to find features and finds.

5.1 MPT -The Map Populating Tool

5.1.1 The Map

The archaeological map will be a changeable grid of two dimensions. The smallest size can be 1x1, and the largest can be as large a map as the system memory can support. A row (01, 02, 03, etc.) and a column(A, B, C, etc.) identify each grid in the square. Each square in the grid has the ability to have a character symbol, an archaeology feature, and one or more archaeological finds. For this first phase, assume the entire archaeological map is covered with vegetation. No stone or ditch is visible on the map.

5.1.2 A Square on the map

Each grid square on the map has a row and column location, a current symbol, and a set of properties detectable using archaeological techniques. To do this, each square has to have:

- A feature type
    - A stone
    - A post hole or pit
    - A natural surface
- Excavation status
    - The square has been dug
    - The square has not been dug
- The current symbol for the square (see Table 1 below)
- Three collections (ArrayLists) of each type of find still buried in the square. (For a list of finds, see Table 2 below)

Table 1: Map Symbols

Letter	Meaning	Properties
Y	Parched, Yellow vegetation, low chlorophyll	A hidden stone
G	Wet, Bright green vegetation, high chlorophyll	A hidden post hole
N	Natural, Natural green vegetation, normal chlorophyll	Nothing hidden
S	Exposed Stone	A stone that has been excavated or already on the surface
P	Exposed Post Hole	A post hole or pit that has been excavated or already on the surface
D	Exposed Natural Dirt	A natural surface that has been excavated or dirt already on the surface

Table 2: List of possible finds

Type of Find	Sensor	Science
Pottery	No reaction	Style Date
Charcoal	Magnetometer	Carbon Date
Metal Object	Metal Detector	Style Date

5.1.3 Required Functions for the MPT

Viewing Option

There should be a viewing option to show the current map to the user. When the map is in memory, either created or loaded, the user should be able to see the current map. For example, a map that shows a buried building, with all the vegetation and place and the standard symbols, looks like Figure 1.

```
  |ABCDEFGHIJKLMNOPQRSTUVWX
--+------------------------
01|gggggggggggggggggggggggg
02|gGGGGGGGGGGGGGGGGGGGGGGg
03|gGggggggggggggggggggggGg
04|gGgYYYYYYYYYYYYYYYYYYgGg
05|gGgYggggggggggggggggYgGg
06|gGgYggggggggggggggggYgGg
07|gGgYggYYYYYYYYYYYYggYgGg
08|gGgYggYggggggggggYggYgGg
09|gGgYYYYggggggggggYYYYgGg
10|gGggggggggggggggggggggGg
11|gGGGGGGGGGGGGGGGGGGGGGGg
12|gggggggggggggggggggggggg
```

Figure 1 Example Map- An unexcavated Roman villa

Set Viewable Symbol Option

While the map in Figure 1 is correct, as a human being, it is uncomfortable to read. Create a submenu that allows the user to set each symbol type. . For example, with the proper substations for natural, parched, and wet symbols, the working map will look like Figure 2.

```
  |ABCDEFGHIJKLMNOPQRSTUVWX
--+------------------------
01|........................
02|.**********************.
03|.*....................*.
04|.*.##################.*.
05|.*.#................#.*.
06|.*.#................#.*.
07|.*.#..############..#.*.
08|.*.#..#..........#..#.*.
09|.*.####..........####.*.
10|.*....................*.
11|.**********************.
12|........................
```

Figure 2 Example Map with substituted symbols

Create the Map

The first task to complete is to create a map of row by column of a size given by the user. Every square on the map should default to being a natural, covered in vegetation marker first (‘g’) in the constructor.

Create and Set Features in Squares

From there, the user should have two selections. Either to change a single square on the map by selecting row and column, or the ability to select an entire row. For the single square, the user should be prompted to enter the row and column, and then a list of the surface symbol for that square.

Create and Set Finds in Squares

Every square can have one or more of each of the three available finds.t   Each square can contain a collection of pottery, a collection of charcoal, and a collection of metal items.

Each of these finds has a date. Pottery and metal work have a dating system based on style and materials. Charcoal, on the other hand, will have a radio carbon date. Prompt the user to enter the type of find and the date, andthen add that find in the appropriate collection.

Save the file

Each map has to be saved to a file. At this stage, use a comma separated value system. The first line of the file should be the number of columns and the number of rows. Each subsequent line of the file will consist of the column, the row, the type of feature, if it is hidden or visible, then the number of pottery sherds, a list of the sherds dates, the number of charcoal, a list of charcoal dates, then the number of metal objects, and a list of metal objects dates.

Example:

```
24,12
..
B,6,S,FALSE,2,400,410,2,407,405,2,400,400
…
```

The first line has the columns and rows of the map.
The second row shows that the at Column B, Row 6, is Stone (S) that is unexcavated. The natural surface would be N, and a pit would be P. This is followed by 2 pieces of pottery at dates 400 and 410. This is followed by 2 pieces of charcoal with dates 407 and 405. This is then followed by a list of 2 pieces of metal work, dated 400 and 400. All told, there would be 288 square definitions in the file.

Load the file

Each map has to be loaded from a file. At this stage, use a comma separated value system from the "Save the file" The image needs to be loaded back into the system.

5.2 The Archaeological Dig Tool (ADT)

Now that a map has been created, the second system may be employed to run an excavation on the map. The two phases of excavation are surveying and digging, and there are a set of tools for each. For the survey, the tools are the visible light spectrum (your eyes), the magnetometer which can find signals from burning, and a metal detector that can alert if there is a metalic find in the ground. For excavation, the tool is a shovel. When a map square has been dug, it is no longer hidden and the relevant feature (Stone, dirt, or Pit) and an accurate count can be made of each type of find in the square: one for pottery, one for charcoal, and one for metal.

5.2.1 Required Functions for the ADT

Load File

The ADT must be able to load in the file from the MPT so to create a working data set for the ADT. This load file functionality should be the same as the routine created for the MPT.

Viewing Option

There should be a viewing option to show the current map to the user. When the map is in memory, either created or loaded, the user should be able to see the current map. For example, a map that shows a buried building, with all the vegetation and place and the standard symbols, looks like Figure 1. This is code resuse from the MPT

Set Viewable Symbol Option

While the map in Figure 1 is correct, as a human being, it is uncomfortable to read. A submenu must exist to allow the user to change the current symbol on the chart, without changing the values stored in memory. For example, with the proper substations for natural, parched, and wet symbols, the working map will look like Figure 2. This is code reuse from the MPT.

Search with a Tool

Create a menu that allows you to select a square on the map, and then test the square with a magnetometer, a metal detector, or the visible spectrum. If the map square contains a metal find, set the metal detector value for that square to true. If the map square contains a charcoal find, set the magnetometer for that square to true. The visible spectrum will be handled by the map symbol. See Figure 3.

```
  |ABCDEFGHIJKLMNOPQRSTUVWX
--+------------------------
01|
02|
03|
04|
05|
06|
07|
08|
09|                FFFTTFFF
10|                TTFFFFFF
11|                TTFFFTFF
12|
```

Figure 3 Example Map- An Roman villa with areas scanned for metal items. T for detected, F for nothing, and blank for a square not scanned.

Dig with a Tool

The second stage after surveying the sight is to dig the square. When a square has been selected in this function, the square will no longer be marked hidden. The surface symbol must change accordingly. For example, if the southwest corner of the map is fully excavated, the map will change to Figure 4.

```
  |ABCDEFGHIJKLMNOPQRSTUVWX
--+------------------------
01|gggggggggggggggggggggggg
02|gGGGGGGGGGGGGGGGGGGGGGGg
03|gGggggggggggggggggggggGg
04|gGgYYYYYYYYYYYYYYYYYYgGg
05|gGgYggggggggggggggggYgGg
06|gGgYggggggggggggggggYgGg
07|gGgYggYYYYYYSSSSSSDDSDPD
08|gGgYggYgggggDDDDDSDDSDPD
09|gGgYYYYgggggDDDDDSSSSDPD
10|gGggggggggggDDDDDDDDDDPD
11|gGGGGGGGGGGGPPPPPPPPPPPD
12|ggggggggggggDDDDDDDDDDDD
```

Figure 4 Example Map- An Roman villa with the southwest quadrant excavated.

```
  |ABCDEFGHIJKLMNOPQRSTUVWX
--+------------------------
01|........................
02|.**********************.
03|.*....................*.
04|.*.##################.*.
05|.*.#................#.*.
06|.*.#................#.*.
07|.*.#..######$$$$$$,,$,-,
08|.*.#..#.....,,,,,$,,$,-,
09|.*.####.....,,,,,$$$$,-,
10|.*..........,,,,,,,,,,-,
11|.***********-----------,
12|............,,,,,,,,,,,,
```

Figure 5 Excavated Map with substituted symbols

Viewing Option, Find Counts

In addition, any square that has been dug can be used to create a map of the number of finds. If no finds have been found, that square on the map will be blank. Otherwise, it should contain the count of the number of finds. There should be a map for pottery sherds, a map for metal finds, and a map for charcoal finds. See Figure 6.

```
  |ABCDEFGHIJKLMNOPQRSTUVWX
--+------------------------
01|
02|
03|
04|
05|
06|
07|           2222224521000
08|           1111134100000
09|           0000000021000
10|           0000000000000
11|           0101010101000
12|           0000000000000
```

Figure 6 Excavated Map Showing Pottery Counts

Find the Date

Every find has an associated date. Take the average of all dates of all finds discovered for all square on the map, and work out an average. List this as average minus standard deviation to average plus a standard deviation.

Export Materials

As before, the system must be able to save the information. Each of the maps has to be sent out as a text file for future use. This is not a CSV file, but rather the same image from the screen modes saved as several different files:

- Visible - Map Symbols
- Magnetometer – T and F for charcoal
- Metal Detector – T and F for Metal Finds
- Pottery – Count for each square for pottery
- Metal –Count for each square for metal
- Charcoal – Count for each square for charcoal

6 Comments:

For this project, all participants are expected to put Javadocs in their source code. Systems like Eclipse have friendly built-in auto-formatters. For these projects we expect the following:

Author tags must be placed at the beginning of each class, at the class definition. You must also include a short description of class at the beginning.

The parameter, return, and exception (or throw) tags must be placed at the beginning of each method.

You must include a short description of the function. A get or set routine would only have maybe one phrase. A key method would have a long description. (for example, a file handling routine would require more detail).

The other Javadoc tags will be optional.

e.g.

```
/**
 * This class sets the distance member variable
 * @param newDistance the new distance
 * @return true if successful
 */
```

- Table of Javadoc Tags

Tag	Description	Syntax
@author	Adds the author of a class.	@author name-text
{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	{@code text}
{@docRoot}	Represents the relative path to the generated document's root directory from any generated page	{@docRoot}
@deprecated	Adds a comment indicating that this API should no longer be used.	@deprecated deprecated-text
@exception	Adds a **Throws** subheading to the generated documentation, with the class-name and description text.	@exception class-name description
{@inheritDoc}	Inherits a comment from the **nearest** inheritable class or implementable interface	Inherits a comment from the immediate surperclass.
{@link}	Inserts an in-line link with visible text label that points to the documentation for the specified package, class or member name of a referenced class. T	{@link package.class#member label}
{@linkplain}	Identical to {@link}, except the link's label is displayed in plain text than code font.	{@linkplain package.class#member label}
@param	Adds a parameter with the specified parameter-name followed by the specified description to the "Parameters" section.	@param parameter-name description
@return	Adds a "Returns" section with the description text.	@return description
@see	Adds a "See Also" heading with a link or text entry that points to reference.	@see reference
@serial	Used in the doc comment for a default serializable field.	@serial field-description | include | exclude
@serialData	Documents the data written by the writeObject( ) or writeExternal( ) methods	@serialData data-description
@serialField	Documents an ObjectStreamField component.	@serialField field-name field-type field-description
@since	Adds a "Since" heading with the specified since-text to the generated documentation.	@since release
@throws	The @throws and @exception tags are synonyms.	@throws class-name description
{@value}	When {@value} is used in the doc comment of a static field, it displays the value of that constant:	{@value package.class#field}
@version	Adds a "Version" subheading with the specified version-text to the generated docs when the -version option is used.	@version version-text

-