Acknowledgement

Thanks to the Section Leaders, who helped out so much this week while I was sick! This project was almost entirely invented by them - and they wrote most of the spec, too!

1 Introduction

In this project you will be creating a program which will simulate an ant chasing your mouse on a 2-D grid. This is no ordinary ant chasing your mouse! Each square on the screen has a different color, representing its state. Each time that the ant steps on a tile, it changes the color - but the (old) color also affects how the ant will move next.

Key ideas that will be practiced in this assignment are using 2-D arrays, methods and the StdDraw.mouseX() and StdDraw.mouseY().

2 Ant.java Overview

In your main() method, you will have an infinite loop. Each pass of the loop will represent one “step” of the ant. Each time that the ant takes a step, you will need to do several things:

- Update the position of the ant
- Update the tile that the ant steps on
- If the user is clicking the mouse, set that one tile to ‘red’
- Redraw the entire screen

You will use a 2D array of ints to store the state of all of the tiles on the screen. You will use methods to implement the various parts of your program, so that main() will be simpler.

2.0 New StdDraw Methods

In this project, you will be using four new methods in the StdDraw library:
\textit{StdDraw.mouseX()}

Returns the current x-position of the mouse as a double.

\textit{StdDraw.mouseY()}

Returns the y-position of the mouse as a double

\textit{StdDraw.mousePressed()}

Returns true if mouse button is currently being clicked, false otherwise

\textit{StdDraw.show(int)}:

Displays the current drawing to screen, and then waits the specified number of milliseconds.

\subsection*{2.1 Program Requirements}

- Implement all methods as stated below
- Have the following 4 states for your grid:
  - White: Ant moves towards the mouse
  - Blue: Ant moves away from the mouse
  - Green: Ant moves in a random direction
  - Red: Ant does not move at all
- Correctly implement logic to control the ant’s movement based on the tile state
- Correct change the states of tile as ant moves around
  - Tiles should go from White to Blue, Blue to Green, Green to White, Red to White
- When the user clicks the mouse change the state of the tile to a red tile

\subsection*{2.2 About the methods}

Your program must implement the following methods:

\begin{verbatim}
public static int moveAnt(int gridState, int antCoord, int mouseCoord)
  // gridState is is the state of the grid square where the ant is currently standing
  // antCoord is either the x or y value of the ant’s position on the grid
  // mouseCoord is the current x or y value of the mouse

  This method should return an int which will be the new x or y position of the ant based on the state the tile was in. The ant may move zero steps, or 1 step in either direction.
\end{verbatim}
Note that this method only can update a single coordinate - but you need to update every clock tick!

```java
public static void changeGridState(int[][] grid, int x, int y)
    ● int [][] grid is the 2-D array the ant is walking on
    ● int x and int y are the indices of the tile in grid that should be changed
```

This method will modify exactly one tile in the array. It will advance the state to its next state.

How will you represent four colors using integers?

```java
public static void drawBoard(int[][] grid)
    ● Given a 2-D array this method should draw the entire grid with the correct colors for each tile based on the tile’s current state
```

This method should not call either StdDraw.setScale(int,int) or StdDraw.clear(); these should both be handled by main().

```java
public static void drawAnt(int x, int y)
    ● This method should draw the ant at the position given by x and y
    ● The ant should be easy to distinguish from the tiles on the board
```

At a minimum, you are only required to use a single drawing element (such as a square or circle). However, you are encouraged to be creative! Have some fun with your ant drawing. The only thing you have to do in main to draw your super fancy ant is call this method!

This method should not call either StdDraw.setScale(int,int) or StdDraw.clear(); these should both be handled by main().

### 2.3 Regarding main()

Your main() method is the core of your program. It should include the infinite loop, the required variable declarations, and also the basic setup code (like StdDraw.setScale(int,int)). However, the vast majority of its work should be accomplished by the other methods - keep main() as simple as possible.

In this project, we will be using both StdDraw.clear() and StdDraw.show(int). Always call clear() before you start drawing again - and always call show(int) when you are ready to display it to screen. You get to choose how long to make StdDraw wait after it shows each frame of
animation; if you pass `show(int)` a small value, then the program will run **very** quickly (because it waits only a short time each frame); if you pass `show(int)` a large value, then it will be slow.

### 2.4 About the array
This spec states that you must use a 2D array of `int`. However, it doesn’t specify what size the array must be; it also doesn’t tell you how to encode the colors as integers. You get to choose the size - but make sure that your `StdDraw` scale matches the array size. You will have to decide how to encode the states as integers.

### 2.5 The Edges of the Window
You get to choose what happens to the ant if it reaches the edge of the window. Does the program end? Is the ant blocked from moving any further? Is it allowed to move off-screen? Does it wrap around? Is it warped back to the middle? It’s up to you. However, you must write code to handle this situation - otherwise, you will eventually encounter `ArrayIndexOutOfBoundsException` when the ant reaches the edge.

In the same way, your code which checks to see if the mouse is currently being clicked must be aware of the edges. With `StdDraw`, it is sometimes possible to register mouse clicks which are just a hair past the edge of the screen - what should you do in that situation?

### 3 Pseudocode

Download the latest pseudocode requirements from:

http://www.cs.arizona.edu/classes/cs127a/spring16/projects/

Project 5 will be graded based on v1.2 of the requirements.

When you write the pseudocode for these programs, you don’t need to describe how you will get the inputs from the command line - you’ll do that in the Java code. You may simply assume that you have the right number of variables, which hold the inputs, when your pseudocode begins. (Make it clear what their names are.)

For hints about writing pseudocode, check the Project 1 spec.

### 3.1 Pseudocode and Methods

The new version of the Pseudocode Requirements document shows how to declare a method in pseudocode - and also how to call it. Make sure you read the new information (it’s at the end of the document).
4 Java code

After you complete the pseudocode (and turn it in!), start working on translating it to Java. (For a list of steps for converting the pseudocode to Java, see the Project 1 spec.)

Your class must have exactly the names given above: Ant

Make sure that your Java files are named to match.

4.1 Header Comment

Every Java file that you submit must have a Header Comment - which is a simple comment, at the top of your file, which gives basic information about it. A header comment must include:

- The name of the Java class
- The name of this class (“CSc 127A Spring 16”)
- The assignment name
- Your name
- Your Section Leader’s name (or letter)
- A quick description of the Java class.

Here’s my suggested header comment for Program 5:

/*
* class Ant
*
* CSc 127A Spring 16, Project 05
*
* Author: TODO: your name here
* SL Name: TODO: the name of your SL here
*
* ---
*
* TODO: put a quick (2-3 sentence) description of the class here
*/

5 Turning in Your Programs

See the class website for information about how to upload your files to lectura, and how to use the ‘turnin’ utility.

5.1 Pseudocode
Your pseudocode file must be either TXT or PDF files (one per program), but you may name them whatever you want. You must turn them in using the assignment name

    cs127a_s16_sX_proj05_pseudocode

(replace the X with your section letter). REMEMBER: The pseudocode is due two days before the Java code!

5.2 Java Code

You must turn in file(s) named exactly this:

    Ant.java

You must turn in the Java file(s) using the assignment name

    cs127a_s16_sX_proj05

(replace the X with your section letter)

Late Day Clarifications

Pseudocode

The syllabus says that you can use Late Days for late projects - however, the syllabus does not explicitly say how this works with pseudocode. The SLs and I talked it over, and we decided on the following policy:

- Late days cannot be spent on pseudocode. Turn it in on time, or you’ll just miss that part of the project grade.
- You must turn in your pseudocode using turnin; email will no longer be accepted.

(We were flexible about both of these points in the first two projects - but starting with Project 3, we’re going to enforce this.)

Java code

Starting with Project 3, if you email us your code (instead of using turnin), it will count as a Late Day, even if you do it before the deadline. **Please use turnin.** And remember: if you are having trouble connecting from home, it’s always possible to come to the 228 lab, and upload your file using a thumbdrive.