CS100J Spring 2005 Assignment A6 Rat Race Due 23:59, Sunday, 24 April

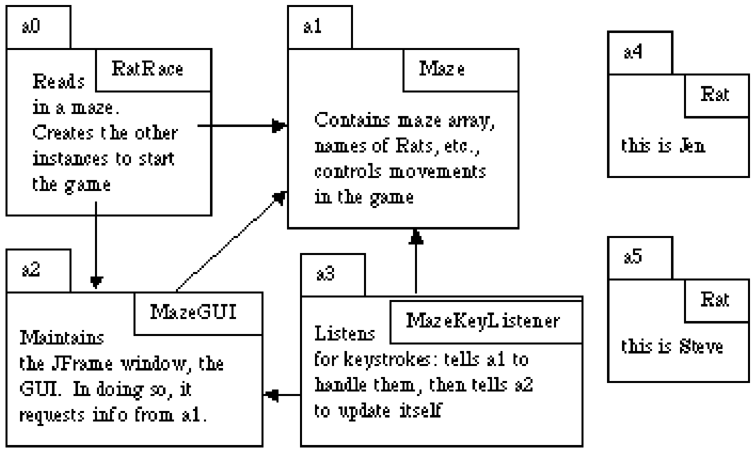
This is the second part of the rat-race game. Spend time reading this handout so that you thoroughly understand what we are asking for. Do this *before* you start programming! Take notes as you read.

If part of this handout is unclear, then please post a message to CS100J news group.

Overview of the game

For an overview of the game, please look at the handout for assignment A5 (in pdf form).

The classes


a0
RatRace
Reads in a maze. Creates the other instances to start the game
a1
Maze
Contains maze array, names of Rats, etc., controls movements in the game
a4
Rat
this is Jen
a5
Rat
this is Steve
a2
MazeGUI
Maintains the JFrame window, the GUI. In doing so, it requests info from a1.
a3
MazeKeyListener
Listens for keystrokes: tells a1 to handle them, then tells a2 to update itself

The program uses five classes, as indicated in the diagram on the right. Class RatRace is used to read in the initial maze and create the necessary instances of the rest of them.

Class Maze is the major calculator. It maintains the maze, it keeps track of the two Rats and the number of sprouts still left, and it is the only one to actually change the maze because of keystrokes. It is the engine. It is important that Maze knows *nothing* about the GUI or the key listener (see below). All it does is keep track of the maze.

Class MazeGUI is an extension of JFrame. An instance of this class maintains the GUI. It has a procedure update(), which is called when the GUI has to be updated; in turn, this procedure calls Maze instance a1 for information it needs (e.g. how many Brusssel Sprouts are in the maze).

Class MazeKeyListener is a new kind of animal for you. It "listens" for keystrokes on the keyboard. When there is a keystroke, its method keyTyped(k) is called by the system. This method figures out what key was typed and calls a method of Maze instance a1 to handle the keystroke; after that, it calls a2.update() in order to change the GUI.

As you can see, each class, or instance of the class, has its own task to do. In this manner, each class can be written fairly easily.

Skeletons for four of the files.

The skeletons have stubs for all the methods that you will have to write in this assignment or the next: RatRace.java Maze.java MazeGUI.java MazeKeyListener.java

Assignment A5

In assignment A5, you wrote a class Rat. Then you wrote method getMap of class RatRace. We assume that these are working.

Task 1

You first task is to write and test all the methods of class Maze. As mentioned above, class Maze contains the methods that manipulate the internal representation of the maze, including the positions of the two rats. Besides the constructor, there are 6 other methods bodies to write. Most of these are simple.

Important point: Note that parameter b of the constructor is a rectangular array that contains 'S' and 'J' to mark the position of the two rats. However, when this maze is stored in field b, the 'S' and 'J should NOT be put in b --they should be replaced by HALL. Instead, the positions of the rats are given by the two Rat folders JenTheRat and SteveTheRat.

Important point. Do NOT use the character constants ".", "@", etc. in writing the method bodies. Instead, rely on the static constants HALL, SPROUT, etc. The same goes for directions. Don't use integers like -1 and +1 to indicate directions. Instead, use the static constants LEFT, RIGHT, NOCHANGE, UP, and DOWN. Points will be deducted if you don't follow these rules.

Task 2

You second task is to write the body of method keyTyped of class MazeKeyListener. This method is called when the JFrame window has the focus and a key is typed. This method should determine what key was typed and perform appropriately. If it is one of the keys (wasdijkl) that are part of the game, the method should call maze.move appropriately. Note that this method has access to the two rats (using maze.JenTheRat and maze.SteveTheRat) and also to the static constants of Maze, like Maze.RIGHT. It should make use of the constants and not use integer constants directly, like -1 or 0 or 1.

Once this method is working properly, you will be able to play the game with your friends. Have a good time with it.

What to submit

On the CMS, submit your files Maze.java, MazeKeyListener.java, Rat.java, and RatRace.java by midnight, Sunday, 24 April.