Rat Crossing Desert

# Instructions

Create a program that finds a path across the desert grid using rats.

General behavior and specifications of program.

The program will consist of one static class: ControlDriver class, and five object producing classes: Desert class, Cell class, Hole class, RatStatus class, and Rat class. There will also be a static ObjectCreator class that I will provide.

The ObjectCreator class is a static class that will provide Desert, Rat, RatStatus, Cell, and Hole objects when requested. Return values from these requests are interface types. The purpose of this class is to hide the production of objects from the client classes. Client classes will only use interfaces to work with objects.

The ControlDriver class is a static class that will

* contain the main() method, which is the entry point into the program
* control the timing of the rat moves.
  + It will control the timing of rat movement across the desert by telling the Desert object when a rat can move. ControlDriver will use a loop to notify the Desert to move a rat for each move.
* only communicate with the Desert and RatStatus objects through their respective interfaces.
* be responsible for initiating the creation of new Rat objects.
  + It will call startRat():RatStatusInterface - Desert and receive back a RatStatus object that contains the unique ID and status for the new rat.
  + it will maintain a list of current rat IDs.
* request the creation of the Desert object
  + It will request the creation of a single Desert object by calling ObjectCreator.createNewDesert(), and work through the DesertInterface of the object in running rats through the desert grid.

The Desert will

* implement the DesertInterface
* create the 2-dimensional CellInterface array for the desert during the construction of a new Desert object.
  + The Desert constructor will query the user for the size of the desert array.
  + The Desert will create a two-dimensional array to represent the desert. It will be of type CellInterface[][], and will hold Cell and Hole objects.
    - Cell class implements CellInterface,
* call ObjectCreator.ceateNewRat() to create a Rat object when prompted and return a unique ID tuple for that Rat object (RatStatus). Note this is not the reference to the Rat object (see Ratstatus class).
* start the rat in location (0,0) of the Desert’s array by doing the following task.
  + will hand over a Rat to a cell and remove the Rat’s reference from its own records.
* be responsible for keeping ***track*** of the rat’s location and movement path(Note: You may delegate keeping track of the rat’s movement path to the Rat. We will discuss this during lecture.)
* The Desert, when prompted by the ControlDriver, will:
  + retrieve a Rat from a cell by asking the cell if it has a Rat with a given ID. It receives the Rat’s reference, and the cell will set its storage of that Rat’s reference to null.
  + prompt the rat for a move direction.
  + The Desert will calculate the rat’s next location using the move direction provided by the rat and the current location of the rat.
  + The Desert moves the Rat one cell in a straight line in the direction indicated by the rat.
  + upon completion of the move, the desert will report back to the ControlDriver with the Rat’s ID and state
    - The state is: 0: when the rat is waiting to move, 1: when the Rat is gone(died or escaped), or -1: When the Rat finishes the Desert.
* will work with the rat object indicated by the control driver, by calling methods from the rat, using the RatInterface. The rat is an object of the Rat class.
* keep rats from falling off the edge. The rats will stay in the grid cell closest to the edge or not move if they are at the edge, but it will still count as a move.
* display the final results
  + retrieve information from the cells for displaying the final results.
  + Of how many Rats died of exhaustion.
  + Of how many Rats disappeared down holes.
* All communication with cell objects will be through the CellInterface or the HoleInterface when needed.

The Cell class will:

* implement CellInterface
* store the Cell’s location on the grid. Do this during the constructor.
* “house” a rat between moves by using receiveRat(Rat) to take the rat and then store it.
* randomly wear down or refresh the rat.
* hold rats that die in the cell.
* return rats that die in the cell.
* return a rat when asked, if that one is housed in it.

The Hole class will:

* extend Cell class
* implement HoleInterface
* override the receiveRat() – Cell method to add the ability to transport the Rat or disappear the Rat
* hold list of other holes. (static) Used to select which cell to transport the Rat to.
* contain a count of rats that disappear down its hole.
* randomly transport rats to other Holes(most of the time)
* randomly allow rats to escape(about 10% of the time)
* return the count of rats that escape/disappear down its hole

The Rat class will

* implement the RatInterface
* maintain an exhaustion/refreshment level
* hold its “alive” status
  + Rat is capable of reporting 2 different statuses
    - alive
    - exhausted to death
* provide a (North, Northwest, West, Southwest, South, Southeast, East, Northeast) movement direction when prompted.
* refreshes or wears down by 1 unit when entering a cell(Cell or Hole)( This is an example, you may implement this with different rates of “wear”.)
  + if it wears down below 6 it dies of exhaustion

The RatStatus class will

* implement the RatStatusInterface
* contain the ID and status of a Rat(Status is finished, dead/missing, alive/waiting to move)
* ID cannot be changed(Constant/final)
* status can be changed to match a rat’s current status.

Responsibilities

If the rat tries to leave the maze by exiting one of the edge cells, across the edge, it stays in that cell until it moves again but also wears down or refreshes by one unit. When a rat dies the control driver decides if a new rat must then brave the desert from the beginning location [0][0](cell notifies desert, desert notifies control and control starts the process to create a new Rat). Count how many rats it takes to find a safe path across the desert from beginning to the end. List the path that was successful. The path is provided by the desert which kept track of the rat’s progress or by the rat that successfully crosses the desert. Each cell also has a “grave yard” (ArrayList<RatInteface>) to hold the rats that died while in the custody of that cell. Since all cells inherit the Cell class, the Cell class should implement the storage and the methods that control the Rat graveyard. The Cewll class should also implement the storage and the methods of storing rats between moves. Allow the user to repeat the program with a new desert, choosing the size of the two-dimensional array again. You must provide documentation IPOs, URLs, and algorithms for this program.

I will provide interfaces for you to implement for the Desert, Rat, RatStatus, Cell, and Hole, classes. I will also provide the ObjectCreator class.

I recommend that you first implement the program with just a small array filled with Cell objects and wait to implement wear down or refresh methods until you get the Rat to cross the desert. Then add in the other features one at a time.