Lab – Growing a Grid

Total time: 20 minutes.

In this lab, we will begin to write the class `MyBoundedGrid` which will implement the `Grid` interface and will work with the GridWorld case study. The `Grid` interface is given here for your reference.

`int getNumRows()`
: returns the number of rows, or -1 if this grid is unbounded

`int getNumCols()`
: returns the number of columns, or -1 if this grid is unbounded

`boolean isValid(Location loc)`
: returns true if loc is valid in this grid, false otherwise
Precondition: loc is not null

`E put(Location loc, E obj)`
: puts object `obj` into location `loc` and returns the object that was already there (or null if the location was previously unoccupied)
Precondition: 1) `loc` is a valid location in this grid and 2) obj is not null

`E remove(Location loc)`
: removes the object at `loc` from this grid and returns it (or null if the location is unoccupied)
Precondition: loc is valid in this grid

`E get(Location loc)`
: returns the object at location `loc` or null if `loc` is unoccupied.
Precondition: loc is valid in this grid

`ArrayList<Location> getOccupiedLocations()`
: returns an array list of all of the occupied locations in this grid

Within your group, develop a template for `MyBoundedGrid<E>` which extends the abstract class `AbstractGrid<E>.` Your template should stub out each of the methods, and each method should be completely documented per the style guide. Your class comment must completely describe the class and it must include each member of your team as author. Be prepared to show your completed template to the entire class.