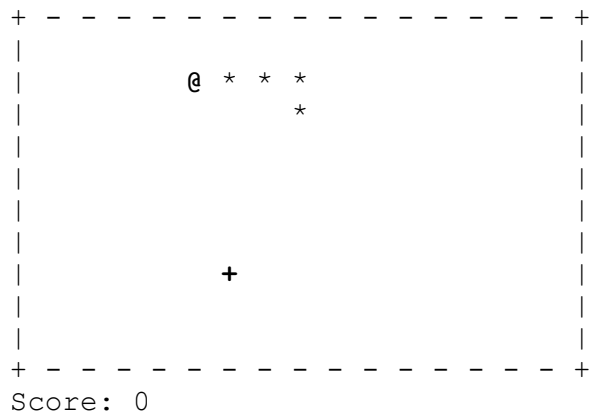
SnakeGame.java

Objective: To use a two-dimensional array and a SinglyLinkedList to create the Snake game.

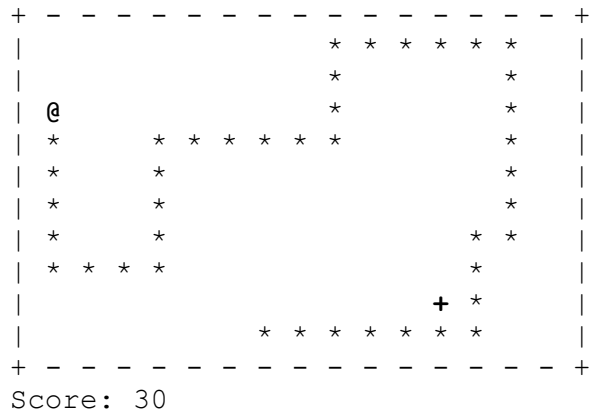
Background:
The Snake Game has been around since the 1970's in one form or another. The first programs were rendered completely with ASCII characters on a terminal, then later with more sophisticated graphics on computers and eventually on handheld devices. The allure of the game is its simplicity in the interface and the rules while being completely addicting to play.

Our game will be modeled on the original ASCII version played on a two-dimensional grid or "board". The "snake" will be a series of characters with the head (@) followed by tail segments (***). An example of the board below shows the snake with a head and a four segment tail.



The user will control the snake and command the head to move up, down, left, or right with the tail trailing behind. The snake can only move to open locations. You cannot have the snake go off the board or cross over its tail; otherwise, you lose immediately (the snake dies).

The objective of the game is for the snake to "eat" as many targets as it can. When a target is eaten, the snake's tail grows by one segment and a new target location is chosen at random on the board. The longer the tail grows, the less open spaces are available, and the more difficult it is for the snake to move. In the figure below, the snake's tail has become so long it is blocking a clear path to the target.



In our game, we will use an **array** to represent the board and a **SinglyLinkedList** to represent the snake.

Coordinate and Snake classes:
You will need to update the **Coordinate** class used in the **SinglyLinkedList** project. A **Coordinate** object in the game holds one location on the board. It has a **row** field and a **column** field, and the constructor initializes those fields. There are also two accessor methods, **getRow()** and **getCol()**, and an **equals()** method.

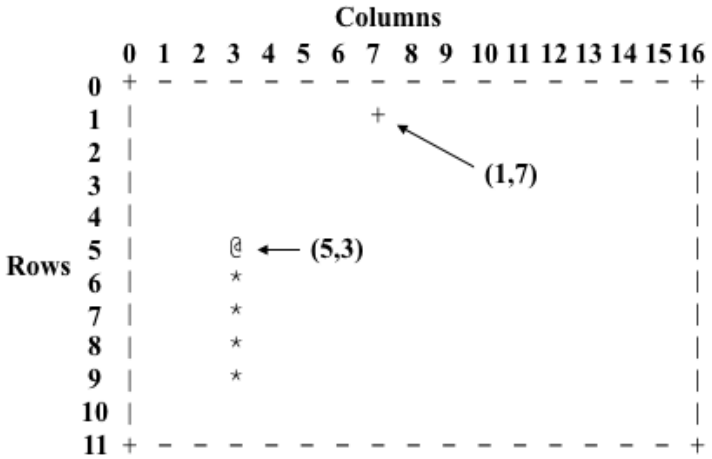
The **Snake** class is a **SinglyLinkedList**. Open the **Snake** file and notice the first line of the class definition.

```
public class Snake extends SinglyLinkedList<Coordinate>
```

The **Snake** class extends the **SinglyLinkedList** class which holds a list of **Coordinate** objects. The **Snake** class inherits all of **SinglyLinkedList**'s methods, so only the constructor(s) needs to be defined. Make the **Snake**'s constructor create a default **Snake** that takes up 5 vertical **Coordinates** on the board.

SnakeBoard class:
The **SnakeBoard** class knows how to construct the board, place a **Snake** and target on the board, and print the board to the screen. The **SnakeBoard** will be stored as a two-dimensional array of **char**'s. The **SnakeGame** class will use the **SnakeBoard** whenever it needs to display the board to the screen.

The **SnakeBoard** contains the border, the **Snake**, and the target. The diagram below shows a board with a 10-by-15 field, the **Snake** (@****), and the target (+).



The head of the **Snake** is designated by the at-sign (@), and the **Snake** always moves head-first. Notice that the row numbers increase from top-to-bottom and column numbers left-to-right. **Coordinates** are in the form (row, column).

Assignment - Coordinate and SnakeBoard:
1) Download the zip file from Mr Greenstein's web site. Unzip the file and it will create a directory **SnakeGame**. Do all of your work in this directory. In the directory you will find the files **Snake.java**, **SnakeBoard.java**, **SnakeGame.java**, and **SnakeGame.jar**.

   The **SnakeGame.jar** file contains a working version of the game using the following command:

   ```
   java -cp SnakeGame.jar SnakeGame
   ```

You will need to copy in the classes **SinglyLinkedList**, **ListNode**, **Prompt**, and **FileUtils**. The remaining files **SnakeBoard.java** and **SnakeGame.java** will be described below.

2) Modify the **Coordinate** class that was used in **SinglyLinkedList**. This class should implement **Comparable** and have two private int fields: **row** and **column**. The constructor inputs and initializes the row and column values. There needs to be two accessor methods, **getRow()** and **getCol(),** a **compareTo()** and an **equals()** method given below. The class should have **compareTo()** and **equals()** methods, and must specify **@Override** annotation for both.

```
@Override
public int compareTo(Object other)
{
      …
}

@Override
public boolean equals(Object other)
{
      return compareTo((Coordinate)other) == 0;
}
```

3) Edit the **SnakeBoard** class given in the zip file. A constructor with the following signature is provided to initialize the width and height of the board.

```
public SnakeBoard(int height, int width)
```

Complete the **printBoard()** method and create as many helper methods as necessary. A main method is provided for testing your **SnakeBoard** class.

SnakeGame class:
The **SnakeGame** class runs the game. It prompts the user and prints the board, it moves the **Snake**, and it randomly places the target on the board. It also knows when to end the game. The main fields are the **SnakeBoard**, the **Snake**, the target, and the score. A skeleton version of the code is provided in the zip file.

The **SnakeGame** knows how the **Snake** works. It moves the **Snake**'s head north, south, east, and west, keeps track of moving its tail in sync with its head, and "grows" its tail. The **SnakeGame** also places the target '+' into random open spaces and keeps track of the score.

SnakeGame declares the game is ended under these conditions:

1. the user quits the game.
2. the snake moves into the border or into itself.
3. the snake's head '@' is surrounded by itself and/or the border and has no place to move.
4. the board has only 5 open spaces. (Maximum score)

SnakeGame provides the following menu of user commands:

w move north
s move south
d move east

a move west
h help
f save game to file
r restore game from file
q quit

Assignment - SnakeGame:

5) Edit the **SnakeGame** class given in the zip file. It needs a constructor to initialize the fields. Initialize the **SnakeBoard** to a size of 20 rows and 25 columns. At the beginning of every game, place the **Snake** and randomly place the target on the board. Remember, the target cannot be placed on top of the **Snake**! You will need the **Prompt** class to accept input from the user.

6) In **SnakeGame**, build a "save and restore game" feature so you can resume the game at a later time. You will need the **FileUtils** class to read from and write to files. Play the game in **SnakeGame.jar** and save it to a file 'f'. This will create a **snakeGameSave.txt** file that you can use as a model for your saved game.