Snake has always been my favorite “simple” game, though I never knew how complicated it could be. I never played Tetris as a kid, but the first cell phone I was ever given had Snake pre-loaded as a default game. I killed countless hours using this simple game, and as such I decided it deserved the tribute to snake that is my project. It was a challenging assignment, because I was creating my game from scratch, versus using the skeleton code for Tetris that came with the default project. I was responsible for designing every aspect of my game. This also however, guaranteed me a fair amount of freedom in my programming.

In Snake, the user controls a small, little, one square snake whose sole purpose in life is to roam around a small box and eat apples. When you consume an apple, you grow by one block. This process continues (forever, potentially) until you either hit the bounding box or collide with another part of the snake.

The most challenging aspect of my project was the snake movement. On the most basic level, the objective is to draw the apple, draw the box, draw the border, draw the snake in its new position, rinse, repeat. This is far more complex than it seems, because the reason the game is referred to as “Snake” is because your character, the snake, needs to slither in snake fashion. In order for this to happen, each body piece(n) must be re-drawn in the position of the body piece in front of it (n-1) each
time the snake moves. Then, after each movement, you must check the position of every body point to ensure that they are not colliding with the snake head, and that the head is not colliding with a wall.

I solved the movement problem using arrays. I created two arrays, piecex and piecey, and a variable length. These arrays could identify what body piece it is, and its location, while length represented the overall body length. I could use the arrays to create the slither illusion with a few crafty for loops:

```java
for(int k=length;k>=1;k--){
    piecex[k]=piecex[k-1];
    piecey[k]=piecey[k-1];
    draw.filledSquare(piecex[k],piecey[k],bodysize);
}
```

This moves the body piece k into the position of k-1 and draws it, piece by piece, creating the movement illusion. After each movement, you simply check the snake head’s position and ensure that it isn’t on the same point as the border or a snake body piece (another loop to check the arrays), and if so, exit out.

At first I was very confused by the GUI, I didn’t know how to go about it. Online tutorials and instructional pages all recommended using JButton and Swing to create a start button, but in truth it was just as efficient and much simpler to just do it within StdDraw. I confused myself by thinking I needed to create a clickable object within the program, when all I had to do was draw a box on the frame with an action listener that detected mouse clicks, and if the user clicks within the box, it chose the appropriate variable values for the speed of the snake. This allows the user to choose their difficulty, and start the game when they’re ready.

Recreating Snake was probably the most interesting project I’ve had in quite some time here at BC. (Also, it has provided me with my own personal procrastination tool for during exam time). It was
entertaining re-engineering my favorite time-waster from middle school while bringing together everything we’ve done in CS1.