CS 158 Lab 9 February 18, 2009
Using Collections and Iterators, part 2

1. Go back into the Chapter 4 project you created in the last lab. In the **Board** class add a static constant called **INITIAL_ASTEROID_SIZE** and set it to 4. In the **Asteroid** class add an integer member variable called **size** and have the constructor set it to the constant you just created. Add a method called **getSize** that does the obvious.

2. In the **applications** package create another class called **Missile**. Add the same member variables that an asteroid has, except for the size variable. Add a constructor that takes arguments for the initial **dx** and **dy**, and sets those instance variables appropriately. In the **Board** class, create public static constants called **INITIAL_MISSILE_X** and **INITIAL_MISSILE_Y** and set them to **SIZE/2** and **SIZE – 10**. Then in the **Missile** class constructor, initialize **x** and **y** to those constants. (This location is down near the bottom of the board in the center. The board is numbered with the origin in the upper left corner, positive x is to the right, and positive y is down.)

3. We want to be able to display asteroids and missiles, but it may take a while before we can develop the right graphics. So we would like to arrange things so that we could use different ways of displaying things, and switch easily when we find something better. To make this possible, we'll start by creating a new package called **gui**, and within this package create an interface called **DisplayObjects**. For this interface, we will define one method (recall that an interface does not contain any implementation of methods – just the signatures). This method is called **displayBoard** and returns nothing, but takes a **Board** argument.

4. In the **gui** package, create a new class called **TextGraphics** that implements the **DisplayObjects** interface. To make this compile, you will have to add a **displayBoard** method. Go ahead and add that method, and we'll fill in the details later. Create a private two-dimensional array of characters in this class called **array**, and instantiate it in the constructor to have size in both dimensions equal to **Board.SIZE**. (In Java, two-dimensional arrays are declared using two sets of brackets: [ ] [ ], with the first set of brackets for the row and the second set of brackets for the column). Create a private method called **init**, and use it to set each element of this array to the character '.' (dot). You will need nested for loops to do this.

5. Go back to the **Board** class and set the size constant to 10. Add a method called **getAsteroids** that returns the entire collection of asteroids. This is not necessarily a good practice, but it will save us a lot of work (having to turn the **Board** class into a real container class).

6. In the **TextGraphics** class we will now fill in the **displayBoard** method. Start by getting from the **Board** argument the collection of asteroids. Call **init**. Then for each asteroid in the collection, get its x and y coordinates, and set the corresponding element of **array** to the character 'a'. After you do this, print on the console the entire array. To test this method, create a new class in the **applications** package called **Game** that contains a **main** method. In the **main** method, create a **Board**, and have the **Board** add 5 asteroids. Declare a **DisplayObjects** variable called **displayer**, and instantiate it as a **TextGraphics** object. Tell the displayer to display the board. Run the main method of the Game class and verify that the board displays the asteroids in random locations.

7. Add a **move** method to the **Board** class that sends a **move** message to each of the asteroids in its collection. In the **Asteroid** constructor, set the initial value for **dx** to 0 and **dy** to 1. Back in the **Game** class, after you display the board, send a **move** message to the board, and then display the board again. Verify that the asteroids are where they are supposed to be.

8. In the **Board** class, add a member variable collection of **Missile** objects similar to the one used for **Asteroid** objects. Instantiate that collection in the constructor. Add a method called **addMissile** that adds a missile to this collection that, for now, has values for **dx** and **dy** that are 0 and 1 respectively. Add a method called **getMissiles** that is similar to **getAsteroids**. Modify the move method so that it also tells every **Missile** in its collection to move. Back in the **Missile** class, add a boolean **move** method that changes the x and y values by their increments, but if the new values end up off the board, returns false, otherwise returns true. Also add get methods **getX** and **getY** that return the missile's x and y location. Go back to the **Board** class and change the **move** method so that when missiles are moved, if the missile's move method returns false, the missile is removed from the collection. Change the **TextGraphics** class so that missile objects are put into the array after asteroids. Finally, run the game to verify that the missile appears on the board and moves appropriately.