Unit 8. GridWorld
Programs.

For simplicity's sake, always include these two import statements:

```
import info.gridworld.actor.*;
import info.gridworld.grid.*;
```

Part 2.
Do the CircleBug, SpiralBug, and ZBug exercises described on pages 4 and 5 of part 2 of the online documentation.

- You should use the BoxBug code as an example.
- Override the act method
- Use only the methods inherited from the Bug class (canMove, move, and turn).
- Add instance variables as needed.

Part 3.
1. A Bug always starts by facing north. Create a Random Bug that starts in any direction. Otherwise it acts exactly like a bug (including leaving flowers behind).

Remember to test this by creating a RandomRunner file (or something similar).

2. Create a new file with a main method. Create an array of 50 Random Bugs and give each one a random color. Remember:

```
int red = (int)(256*Math.random() );
int green = (int)(256*Math.random() );
int blue = (int)(256*Math.random() );
Color col = new Color( red, green, blue );      // this is a random color
```

Add the 50 bugs to random locations in the grid by calling the correct constructor.

3. Create a Fragile Bug. It moves just like a regular bug except if it cannot move (because of a rock or the edge of the grid), it removes it self from the grid.
To find the right method to remove the bug, look at the list of methods that appear when you right click a bug in the grid.

Remember to test this by creating a FragileRunner file (or something similar).

4. Create a Rainbow Bug. It is a subclass of the BoxBug class. Every rainbow bug tries to move in a 4 by 4 box. In addition, it changes color depending on which direction it is headed at the end of each move.

When it is headed north, it is blue.
When it is headed east, it is green.
When it is headed south, it is red.
When it is headed west, it is yellow.

The boxbug class must be in the same project folder as the rainbow class. The requirement that every rainbow bug move in a 4 by 4 box (when possible) will have an impact on how you write the constructor for this class.

5. Create a DyingFlower that acts like a flower except that after it has "acted" six times, it removes itself from the grid.

6. Create a TestBug that acts like a Bug except that it leaves behind dying flowers instead of regular flowers.

7. Create a BorderBug that acts like a Bug except that it is red whenever it is on the first or last row or column. It is yellow whenever it is immediately next to those locations. Otherwise it is green. Do NOT assume that the grid is a 10 by 10 grid.

For example: If the grid has 7 rows and 5 cols, this bug will be green at location (4, 2), yellow at location (5, 3) and red at location (3, 4).

Part 4 (Critters). When extending the critter class:

- Never override the act method
- When overriding one (or more) of the methods that act calls, your method must follow any postconditions of the overridden method.

8. Do the following three exercises from pages 35 and 36.

- ChameleonKid. Look in the Color class for a method that will help you make it darker.
- BlusterCritter.
- KingCrab.

9. Make a RockCritter that behaves like a Critter except, instead of eating any non-Rock/non-Critter neighbors, it will eat all the Rocks that are in its row. In other words, it removes all Rocks from its row and then it moves like a regular critter.

10. Make a HungryHungyCritter that behaves like a Critter except that it eats all non-Rock/non-Critter actors that are with two squares of it (in any direction). In other words, if it is in the middle of a 10 by 10 grid, it will look at 24 locations looking for delicious actors (but not Rocks or Critters) to eat. This thing is a monster.

11. Make a TeleporterCritter. It behaves like a Critter but it can move to any empty cell on the grid. If there are no empty cells it remains in the same location.

12. Make a WrapAroundActor. It extends Actor. Its initial direction is north, east, south or west (randomly selected). It then moves in that direction forever. When it reaches the edge of the grid, its next move will take it to the opposite side of the grid (and still facing the same direction). For example, if it faces east and its location is row 4, column 9 (in a standard 10 by 10 grid), then the next time it moves it will be at row 4, column 0.

Do NOT assume that the grid is 10 by 10. Use the appropriate methods to figure out the number of rows and columns in the grid.

Eat anything that is in your way.

13. Make up something interesting that either extends Actor or Critter.