# COMS W1007 Homework 5

### Submission instructions

All programs must compile and run on CUNIX to receive credit. Submit your electronic files via `http://courseworks.columbia.edu`. Post your archived submission directory to the CLASS FILES section in the appropriate homework directory. We prefer electronic submission of written portions, though you will not be penalized for paper submissions. (Do **not** print out your programs.) Include a separate README text file that explains exactly what each file in your submission is. Your submission should contain your code (.java files), written problems, and your README file. Do not include your compiled .class files or your compiled javadoc, since we will generate those from your code. Place all the files you want to submit into a submission directory with the following naming scheme.

`<your_uni>_hw<number>`

So if my UNI is `uni1234` am submitting homework 5, my directory would be `uni1234_hw5`. Archive your submission directory using zip, tar, or gzip format. In CUNIX, you can tar and gzip with the following command:

```
tar -czvf uni1234_hw5.tar.gz uni1234_hw5
```

and upload `uni1234_hw5.tar.gz` to courseworks. (You will probably need to first download the file to a local directory using an FTP program. See CUNIX tutorial for more info.)

Multiple Submissions: You can submit multiple times, but we will only consider the latest submission based on the timestamp in courseworks. Please give at least 1-2 minutes between two submissions so we can tell which is the newest submission.

Keep a pristine copy of your submission folder in case there is any submission error.

## Written Problems

1. (6 points) **Horstmann 9.8** What happens when a thread calls `await` and no other thread calls `signalAll` or `signal`?

2. (6 points) Suppose two threads are sharing an `LinkedList`. One thread is adding elements to the `LinkedList` and the other is only reading, but never modifying the `LinkedList`. Can a race condition occur in this scenario? If so, give an example? Or, if not, why is this setup safe? What, if anything, is necessary to ensure the `LinkedList` behavior is not corrupted by race conditions?

3. (6 points) The `Collections` class contains a static method `unmodifiableSortedMap` [1]. Read the API, describe what these do and describe how you would implement them (your approach does not need to be exactly how the standard library does it as long as it works).

4. (6 points) **Horstmann 10.2** Explain why `MouseAdapter` and `WindowAdapter` are not adapters in the sense of the ADAPTER pattern.

5. (6 points) **Horstmann 10.6** The STRATEGY and COMMAND patterns both suggest using objects in place of methods. What is the difference between these two patterns?

## Programming Assignment

(70 points) For this assignment, you will build a pigeon feeding simulation, which will use multithreading and simple data structures. The game will simulate pigeons roaming around a public area (e.g., the steps of Low Library). The game takes place in a window on which some "pigeons" will wait for food. Each pigeon must be controlled by a separate thread. The user clicks on a spot on the window and food is immediately placed at the location of the click.

The pigeons will move according to the following rules:

- If there is no food anywhere, each pigeon goes to sleep and does not move.
- If there is any food, each pigeon moves toward the freshest food (the most recently placed food).
- If a pigeon touches the freshest food, it eats the food and the food is deleted.
- If any pigeon accidentally touches food that is not the freshest available, it ignores it and does not eat it. (So you only need to do collision-detection between pigeons and the freshest food.)
- The pigeons will randomly get startled, which causes them to quickly fly to a random location. Choose a constant to determine the probability of this happening each "turn", and when it does, animate the pigeon quickly flying to its new random location. This behavior will reduce the change that the pigeons will converge and all overlap on the screen.

The food and the pigeons should be drawn graphically on the screen, but don't spend too much time on the graphics; you may draw the food and pigeons as simple shapes, such as circles, but the key exercise in this assignment is to make sure the multithreading and data structures work. The multithreading issues you must handle follow.

- You will need to make sure the pigeons stop when there is no food available. The threads should actually stop, which means your program will use very low (almost zero) cpu usage when there is no food available.
- You will also need to make sure that if two pigeons reach the food simultaneously, only one pigeon deletes the food from your data structure.

[1] It also contains similar methods `unmodifiableSet`, `unmodifiableSortedSet`, `unmodifiableList`, `unmodifiableMap`, and `unmodifiableCollection`.

- When your drawing code executes, it will probably need to iterate over an `Iterable` of food locations. Since unsynchronized code allows food to be added or deleted during this drawing code, you should lock the data structure(s) while drawing.

You should choose a data structure that allows your program to efficiently find the freshest food in **constant time**, no matter how much food has been already placed on the ground. There are a couple ways to do this.

Model-view-controller (MVC) is a useful starting-point for this assignment, but you are not required to follow the pattern this time. Nonetheless, it is a good starting point for your design, since the game does involve some data state, a view of that state and some controllers (the user and the pigeon threads). However you structure your data and view classes, make sure to think about how each pigeon thread will move its representation on the screen in a thread-safe way.

Figure 1: Two pigeons wait for you to drop crumbs.

There is some freedom in how you design your pigeon behavior, such as defining the size of their representations on screen, their speed [2], the number of pigeons, how often pigeons get startled, how fast the pigeons fly away when startled, etc. These are all up to you, as long as the core functionality is there and implemented using multithreading.

## Process and Deliverables

For this programming assignment, you are only required to submit code that compiles via javac and javadoc. Also submit the code from Horstmann's framework. At least all public methods must have javadoc comments for any parameters and return values. You should include your code and a "README" file that briefly describes each included file, and can be used to indicate to us any notes that would be helpful for grading (such as any extra functionality you have added or were unable to finish in time).

You are encouraged to use the design tools we practiced on previous homework, but you do not need to submit them.

As always, to get full credit, your code must not crash for any reasonable user behavior. You may exit with an informative error message, but uncaught exceptions or crashes will lose some points.

[2] You may want to add some randomization to the speed and direction of the pigeons so the animation looks more organic.