CSC 300 Spring 2019 -- Lytinen
Midterm Exam Information

The midterm exam is at the following times/locations

- In-class students **Wednesday, May 1 from 1:30-3**, in our regular classroom.
- Online students: **Sometime between May 1 and May 5.** You must register on **D2L** to take the exam. You will choose a 90-minute time period.

The exam is **closed book** and **open notes** (you may bring as many notes as you would like). **Computing devices are not allowed,** including laptops, tablets, cell phones, and calculators. The exam will be a mixture of short answer and short coding problems. Given that it is a hard-copy exam and you are not allowed to use computers, each coding problem will only require a small amount of code (fewer than 10 Java statements), and I will not take off for small syntax errors such as missing semicolons, unbalanced { }, etc.

Topics to be covered on the exam may include:

1. **Java basics**: primitive datatypes; classes and objects; variables; method definitions and method calling; instance vs. class (static) methods and variables; defining and passing parameters; packages; if statements; if…else constructions; loops; arrays.

2. **Advanced Java**: Wrapper classes; inheritance and subtype/supertype relationships; interfaces (especially **Comparable, Iterable, and Iterator**); what it means for a class to implementing an interface; generic classes/interfaces and type variables; Java inner classes (including anonymous inner classes).

3. **Abstract Data Types and Data structures:** The Bag/Collection ADT; various implementation of Bag (array-based, expandable array-based, and Node-based); common operators used on Collections; approximations of running times of operations

4. **Stacks:** (if we get to them) Stack operations (push, pop, peek, isEmpty); array-based vs. node-based implementations; stack applications (e.g., postfix arithmetic; Web browsing history, etc.); operation running times

5. **Choosing a data structure for a particular application** (see practice problem #12 below)

Given the 90-minute time period, it is likely that I will not ask questions that touch upon all of these topics.

Practice problems

There will **not** be this many problems on the exam!!

1. Consider the loop below:

```
int x[] = {'a', 'b', 'c'};
for (int i=0; i<x.length; i++)
  System.out.print(x[i]);
```

   The code prints `abc`. Rewrite the loop as a while loop. Then rewrite it using the "foreach" construction. In both cases, the output should also be `abc`.

2. What is the output of the code below? Explain.

```
public class P2 {
  public static void main(String[] args) {
    int x = 1;
    int y[] = {1, 2};
    f(x, y);
    System.out.println(x + " " + y[0]);
  }

  public static void f(int x, int[] y) {
    x = 10;
    y[0] = 10;
  }
}
```

3. Below are method calls to 3 of the methods of the **String** class. Which are correct, and which are incorrect? Explain your answers.

```
String s = "abc";
char c1  = s.charAt(1);        // (1)
int len  = String.length(s);   // (2)
String s = String.valueOf(1);  // (3)
```

4. Fill in the missing code for the **MyInteger** class below. Specifically, write an **equals** method, and complete the **compareTo** method.

```
// this is a built-in Java interface, but I've
// supplied code for your reference
//
// public interface Comparable<Item> {
//   public int compareTo(Item other);
// }

public class MyInteger implements Comparable<MyInteger> {
  public int x;  // this would normally be private, but I made it public
                 // for the purpose of describing the methods below

  public MyInteger(int x) {
    this.x = x;
  }

  // write an equals method below.
  public boolean equals(MyInteger i) {
```

```
    return false; // maybe replace this
  }

  // Write a compareTo method below.
  public int compareTo(MyInteger i) {




     return 0;  // maybe replace this
  }
}
```

5. When is storage space allocated and de-allocated for each of the following types of variables?
   - local variables
   - instance variables
   - static variables (i.e., class variables)

6. Complete the `Letters` class below. It should be completed in such a way that the program prints

```
a aa aaa aaaa aaaaa
b bb bbb
```

```
import java.util.Iterator;

public class Letters implements Iterable<String> {

  private char letter;
  private int max_length;

  // x is the maximum length String that the Letters class should produce
  public Letters(char c, int x) {
     letter = c;
     max_length = x;
  }

  // fill this in
  public Iterator<String> iterator() {
    // replace this
    return null;
  }

  public static void main(String[ ] args) {
    Letters let = new Letters('a', 5);
    for (String s : let)
      System.out.print(s + " ");
    System.out.println( );
    let = new Letters('b', 3);
```

```
    for (String s : let)
      System.out.print(s + " ");
    System.out.println( );
  }
}
```

6. Write a `main` method which does the following in the order specified:

   a) Creates an empty ExpandleBag of Strings
   b) Adds the Strings "a", "b", and "c" to the bag
   c) Prints the contents of the Bag, without explicitly using the Bag's `toString` method (that is, the code you write may not include a call to `toString`)
   d) Prints the contents of the Bag, without explicity using the Bag's toString method, AND without using the Java "for-each" construction.

9. Characterize the running time ***T(n)*** of each of the operations below for a Node-based **Bag**, based on the amount of data ***n*** that is currently in the Bag. State whether ***T(n)*** is proportional to ***n***, to a constant (e.g., 1), or perhaps some other function of ***n***.

   i. The `contains` method
   ii. The `remove` method

10. Write a Bag method called **count**. It is passed an object called **item** of type T, and counts the number of times **item** is in the **Bag** object. You may assume that the data class (**T**) has an **equals** method.

11. Which data structure (Bag, or Stack) would you choose for each of the following?

    a. You want to implement the "undo" operation (usually CTRL-z) in a text editor.
    b. You are writing a program to determine whether or not a sentence contains any duplicate words. For example, "to be or not to be" contains duplicates (the words "to" and "be"), but "that is the question" does not.

12. A *queue* is a data structure which contains items. It commonly is defined to have the following operations:

    a. enqueue Place a new item onto the back of the queue.
    b. dequeue: Remove an item from the front of the queue.

Queues display **FIFO** (first-in first-out) behavior. For example:

```
Queue<String> q = new Queue<String>();
q.enqueue("abc");
System.out.println(q.dequeue());
q.enqueue("def");
q.enqueue("ghi")
```

```
System.out.println(q.dequeue())
System.out.println(q.dequeue());
```

The output of this code would be

```
abc
def
ghi
```

a. Write an interface which would be appropriate for Queues.
b. If you were asked to implement this interface, would you select an array-based implementation, or a Node-based implementation? Why?