CS 112 Spring 2018 — Introduction to Computer Programming
25 Apr 2018 — Final Exam

Name: _____

NetID (e.g., bb746): _____

My signature below certifies that I have complied with Yale's Academic Regulations and course policy in completing this examination.

_____ _____
Signature Date

Instructions:

- **Do not open this exam until told by the proctor.** You will have exactly 45 minutes to finish it.
- **Make sure your phone is turned OFF (not to vibrate!) before the exam starts.**
- Food, gum, and drinks are strictly forbidden.
- **You may not use your phone or open your bag for any reason**, including to retrieve or put away pens or pencils, **until you have left the exam room**.
- This exam is closed-book, closed-notes, and closed-computational devices.
- If you get stuck on a problem, it may be to your benefit to move on to another question and come back later.
- All code must be written out in proper Java format, including all curly braces and semicolons.
- Do not separate the pages. If a page becomes loose, reattach it with the provided staplers.
- Staple all scratch paper to your exam. Do not take any sheets of paper with you.
- If you require extra paper, please use the backs of the exam pages. Proctors have additional scratch paper if you need more than that. **Clearly indicate on the question page where the graders can find the remainder of your work (e.g., "back of page" or "on extra sheet")**.
- Use any color(s) pen or pencil **except red or pink** to complete the exam.
- If you have any questions, raise your hand and a proctor will come to answer them.
- When you turn in your exam, you may be required to show ID. **If you forgot to bring your ID, talk to an exam proctor immediately**.
- Good luck!

Scores: [For instructor use only]

Question 1		1 pts
Question 2		15 pts
Question 3		11 pts
Question 4		18 pts
Question 5		30 pts
Total:		75 pts

1.) The Easy One (1 point total)

- Check to make certain that your exam has all 5 pages (excluding the cover sheet).
- Write your name and NetID on the front of the exam.
- Sign the certification that you comply with Yale's Academic Regulations and course policy.

2.) What's the Point? (15 points total)

Fill in the code for each of the functions below.

```
public class Point {
    private double x, y; // coordinate of point

    public Point(double x, double y) { // create point with the specified coordinates




    }

    public double getX() { // return the x coordinate


    }

    public double getY() { // return the y coordinate


    }

    // return a point halfway between this point and p (average of the x's and y's)
    public Point midpoint(Point p) {


    }

    // test all the class's public methods (max 5 lines of code!)
    public static void main(String[] args) {









    }
}
```

3.) Objects (11 points total)
For each question below, circle every correct answer (there may be more than one)

3.1) (2.5 points) Why are instance variables commonly declared as `private`?

(a) Users of the class shouldn't care about an object's internal representation

(b) They haven't been promoted to corporal

(c) Users of the class shouldn't be allowed to directly manipulate an object's internal representation

(d) Objects should usually have their own private copies of each instance variable, instead of every object sharing the same variable

(e) None of the above

3.2) (3 points) How do constructors **differ** from all other methods?

(a) Constructors don't have to have a return statement

(b) Constructors don't have a return type

(c) Constructors can't be called directly like other methods

(d) A class can have multiple constructors as long as the signatures differ

(e) A class can have multiple constructors as long as the return types differ

(f) None of the above

3.3) (3 points) Which of the following declarations could be correct for `Integer.parseInt()` (which is defined in the `Integer` class somewhere in the bowels of Java)?

(a) `public static void Integer.parseInt(String[] args)`

(b) `public static int parseInt(String[] args)`

(c) `public void Integer.parseInt(String[] args)`

(d) `public int parseInt(String[] args)`

(e) `public int parseInt(int n)`

(f) None of the above

3.4) (2.5 points) You're debugging your (object-oriented) program. You find that the issue is when instances of a class `C` update one of the instance variables `var`, the same variable is changed in the rest of the instances. Furthermore, this variable always maintains the same value across all instances of the class. What are possible causes of this problem?

(a) The instance variable is declared `public` instead of `private`

(b) The instance variable is declared `static`

(c) `var` is an object type, and all instance of `var` refer to the same object

(d) The instance variables are secretly colluding in violation of the Sherman Antitrust Act

(e) None of the above

4.) Mystery LL Program (18 points total)

Use the `LL` program at the end of the exam (feel free to tear out the page) to answer the questions below. If the program crashes with the arguments listed in any of the questions, write the output that the program prints out followed by an approximate error message. Don't worry about phrasing the error message just like Java, as long as you use **at most 5 words** and show you understand what the error is.

We recommend drawing out the linked lists as you trace through the code.

4.1) (4 points) What is the output of running "`java LL`" with no arguments?

4.2) (4 points) What is the output of running "`java LL 1`"?

4.3) (4 points) What is the output of running "`java LL 1 2`"?

4.4) (4 points) What is the output of running "`java LL 1 2 3`"?

4.5) (2 points) What is a good name for the `mystery` method?

5.) A Steaming Pile of Shrimp (30 points total)

Dim sum restaurants usually offer many different steamed shrimp dishes, which are generally served from carts. Servers stack many steamer baskets of each dish, one on top of another, and they offer the top basket from each stack to patrons as they circulate through the restaurant. Of course there is a maximum height to any given stack before it will tip over. Your job is to write a `DimSum` class that models a stack of `SteamerBasket` objects and can be used to create many different stacks of baskets. It should contain the following public methods, and you can include any additional private methods and instance variables you require. You must write all the code, including the class declaration. Comments are not required. You do not need to use linked lists.

You do NOT need to write the `SteamerBasket` class. Just assume it exists and write the `DimSum` class. If you need more space, please continue on the back of this page.

- `DimSum`: create an empty stack that can hold at most `n` steamer baskets; throw a `RuntimeException` if `n` is less than 1
- `isEmpty`: return `true` if this stack is empty, or `false` if it is not
- `add`: add another `SteamerBasket` to the top of the stack; throw a `RuntimeException` if the stack is already full or the basket to add is `null`
- `serve()`: remove a `SteamerBasket` from the top of the stack and return it; throw a `RuntimeException` if the stack is empty

Mystery LL Program Code

```
public class LL {
    public int val;
    public LL  next;

    public LL(int v, LL n) {
        val  = v;
        next = n;
    }

    // Remember that Java calls this automatically
    // whenever it needs to convert an LL to a String
    public String toString() {
        String s = "" +  val;
        if (next != null) s += " -> " + next;
        return s;
    }

    public LL mystery() {
        if (next == null) return this;

        LL newHead = next.mystery();
        next.next = this;
        next = null;
        return newHead;
    }

    public static void main(String[] args) {
        LL node = null;

        for (int i = 0; i < args.length; i++)
            node = new LL(Integer.parseInt(args[i]), node);

        System.out.println(node); // print out the linked list

        node = node.mystery();

        System.out.println(node); // print out the linked list again
    }
}
```