CS 61B Spring 2024

Midterm 1

Thursday February 15, 2024

Intro: Welcome to CS61B Midterm 1!

Your Name: _____

Your SID: _____ Location: _____

SID of Person to your Left: _____ Right: _____

Formatting:

- ◯ indicates only one circle should be filled in. ☐ indicates more than one box may be filled in. **Please fill in the shape completely.** If you change your response, **erase as completely as possible**.
- Anything you write that you ~~cross out~~ will not be graded.
- You may not use ternary operators, lambdas, streams, or multiple assignment.

Tips:

- This midterm is worth **100 points**, and there are a lot of problems on this exam. Work through the ones with which you are comfortable first. Do not get overly captivated by interesting design issues or complex corner cases you're not sure about.
- Not all information provided in a problem may be useful, and you may not need all lines.
- **We will not give credit for solutions that go over the number of provided lines or that fail to follow any restrictions given in the problem statement.**
- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we may deduct points if your answers are much more complicated than necessary.
- Unless otherwise stated, all given code on this exam compiles. All code has been compiled and executed before printing, but in the unlikely event that we do happen to catch any bugs in the exam, we'll announce a fix.

Write the statement below in the same handwriting you will use on the rest of the exam: "I have neither given nor received help on this exam (or quiz), and have rejected any attempt to cheat. If these answers are not my own work, I may be deducted 9,876,543,210 points on the exam."

_____

_____

_____

_____

_____

Signature: _____

1 WWJD

(15 Points)

For each of the following lines, write the result of each statement: Write "CE" if a compiler error is raised on that line, "RE" if a runtime error occurs on that line, "OK" if the line runs properly but doesn't print anything, and the printed result if the line runs properly and prints something. If a line errors, assume that the program continues to run as if that line did not exist. Blank lines will receive no credit.

```
1  public interface Sphere {
2       default void getRadius() { System.out.println(0); }
3  }
4
5  class Planet implements Sphere {
6      private int radius = 5;
7
8      @Override
9      public void getRadius() { System.out.println(this.radius); }
10
11     public void orbit(Planet p) { System.out.println("orbit planet"); }
12 }
13
14 class Exoplanet extends Planet {
15     static int distance = 10;
16
17     public void getDistance() { System.out.println(distance); }
18
19     @Override
20     public void orbit(Planet p) { System.out.println("orbit exoplanet"); }
21 }
```

(a) Fill in the blanks below:

```
Class Main {
    public void main(String[] args){
        Planet kepler = new Exoplanet();
        Sphere jupiter = new Sphere();
        Sphere arion = new Exoplanet();
        Planet earth = new Planet();

        kepler.orbit((Exoplanet) arion);
        ((Exoplanet) earth).orbit(kepler);
        ((Exoplanet) kepler).getDistance();
        earth.getRadius();
        arion.getDistance();
        earth.radius;
        Exoplanet.getDistance();
    }
}
```

1 _____

2 _____

3 _____

4 _____

5 _____

6 _____

7 _____

8 _____

9 _____

10 _____

11 _____

(b) Within the blank below, briefly explain why changing the access modifier on line 2 to, `private default void getRadius()`, would result in a compilation error. Answer in 10 words or less.

(c) Within the class `Exoplanet`, we define a new method below. Which of the following changes will make `Exoplanet` compile?

```
public void fixRadius() { super.radius = 6; }
```

◯ Change the `private` keyword to `public` on line 6.

◯ Modify `fixRadius()` above to be `private`.

◯ Modify `fixRadius()` above to be `static`.

(d) True or false: we can create another interface that extends `Sphere`.

◯ True

◯ False

(e) It is _____ possible to implement two interfaces in the same class.

◯ Always

◯ Sometimes

◯ Never

2 Comparable Computers

(20 Points)

AJ′s computer versions are represented by a string containing non-negative integers where each term is separated by periods, for example: "123.4.5". Implement the `compareTo` method in the `Computer` class to compare computer versions. A term with the higher numerical value is considered greater. The version with the larger, most significant term is considered greater. See examples below for expected behavior.

Hint: Use `Character.getNumericValue(char c)`, which returns the integer value of the char, so e.g. '0' becomes 0.

```
// Test Case 1: Different numerical values
String str1 = "1.2.3";
Computer c1 = new Computer(str1);
String str2 = "1.2.4";
Computer c2 = new Computer(str2);
System.out.println(c1.compareTo(c2)); // Output is less than zero (1.2.3 < 1.2.4)

// Test Case 2: Different lengths
String str3 = "1.0.0";
Computer c3 = new Computer(str3);
String str4 = "1";
Computer c4 = new Computer(str4);
System.out.println(c3.compareTo(c4)); // Output is zero (1.0.0 == 1)

// Test Case 3: Different numerical values
String str5 = "2.0";
Computer c5 = new Computer(str5);
String str6 = "1.999";
Computer c6 = new Computer(str6);
System.out.println(c5.compareTo(c6)); // Output is greater than 0 (2.0 > 1.999)
```

```
public class Computer implements Comparable<Computer> {

    String version;

    public Computer(String version) {
        this.version = version;
    }

    @Override
    public int compareTo(Computer other) {
        int i = 0, j = 0;

        while ( _____1_____ || _____2_____) {
            int num1 = 0, num2 = 0;

            while (i < this.version.length() && _____3_____) {
                char c = this.version.charAt(i)

                num1 = num1 * 10 + _____4_____;
                i += 1;
            }

            while (j < other.version.length() && _____5_____) {
                char c = other.version.charAt(j)

                num2 = num2 * 10 + _____6_____;
                j += 1;
            }

            if (_____7_____) {

                return _____8_____;
            }

            i += 1;
            j += 1;
        }

        return _____9_____;
    }
}
```

3 Connect The Dots

(20 Points)

Justin is playing connect-the-dots! Justin is given a number n and a rectangular 2-D array `paper`, whose elements contain exactly 1 copy of the numbers 1, 2, ..., n, and all remaining cells filled with 0. For example, the following is an example of a valid `paper` with $n = 6$:

0	0	0	2
0	1	0	0
0	0	0	0
0	3	0	0
4	6	0	5

Justin will draw a straight line starting from the cell marked 1 to the cell marked 2, then to the cell marked 3, and so on until reaching the cell marked n. Write a function `lineLength`, which determines the total length of the lines drawn (adjacent cells are distance 1 apart).

In the above example, the lines from 1 to 2, 2 to 3, 3 to 4, 4 to 5, and 5 to 6 are $\sqrt{5}$, $\sqrt{13}$, $\sqrt{2}$, 3, and 2 units long, respectively. This is derived from the Pythagorean theorem – for example, 1 to 2 is 1 unit up, 2 units to the right for a total of $\sqrt{1^2 + 2^2} = \sqrt{5}$ units along the diagonal.

The total length of these lines is thus $\sqrt{5} + \sqrt{13} + \sqrt{2} + 3 + 2 \approx 12.256$ units. Therefore, if `lineLength` received this `paper` as input, it would return 12.256.

The `Point` class below represents a point in two-dimensional space with integer coordinates and is used in parts (a) and (b) below.

```
1   public class Point {
2       public int x;
3       public int y;
4
5       public Point(int x, int y) {
6           this.x = x;
7           this.y = y;
8       }
9   }
```

(a) Complete the `distance` method below so that it returns the distance between two `Point` objects. You do not need to use all the blanks.

```
public static double distance(Point a, Point b) {

    _____;

    _____;

    _____;
}
```

(b) Implement the `lineLength` method so that it determines to total length of the lines drawn. The method implemented in part a may be helpful.

```
public static double lineLength(int[][] paper, int n) {
    Point[] points = _____1_____;

    for (int i = 0; _____2_____; _____3_____) {

        for (int j = 0; _____4_____; _____5_____) {

            int z = paper[i][j];

            _____6_____;
        }
    }

    double total = 0.0;

    for (int k = _____7_____; _____8_____; _____9_____) {

        _____10_____;
    }
    return total;
}
```

4 Class Signatures

(13 Points)

Fill in the class signatures so that the code below compiles and runs without errors.

```
1  B bb = new B();
2  A aa = new A();
3  A ab = new B();
4  D db = new B();
5  A ac = new C();
6  B bc = new C();
7  A ad = new D();
8  E ed = new D();
9
10 public __(a)__ A {
11     // Code not shown
12 }
13
14 public __(b)__ B ____(f)____ _(g)_ {
15     // Code not shown
16 }
17
18 public __(c)__ C ____(h)____ _(i)_ {
19     // Code not shown
20 }
21
22 public __(d)__ D extends _(j)_ implements _(k)_ {
23     // Code not shown
24 }
25
26 public __(e)__ E {
27     public default void e() {
28         System.out.println("E!");
29     }
30 }
```

(a): ◯ class ◯ interface

(b): ◯ class ◯ interface

(c): ◯ class ◯ interface

(d): ◯ class ◯ interface

(e): ◯ class ◯ interface

(f): ◯ extends ◯ implements

(g): ◯ A ◯ B ◯ C ◯ D ◯ E

(h): ◯ extends ◯ implements

(i): ◯ A ◯ B ◯ C ◯ D ◯ E

(j): ◯ A ◯ B ◯ C ◯ D ◯ E

(k): ◯ A ◯ B ◯ C ◯ D ◯ E

5 HOF

(16 Points)

Consider the following interface, which represents an arbitrary function. `apply` takes an input of type A and returns an output of type B.

```
1  interface Func<A, B> {
2      B apply(A x);   // ... all other methods omitted ...
3  }
```

Complete the `compose` method, which takes in two functions f and g and returns a new function that, when applied, returns $g(f(x))$, or the result of applying f and then applying g. Be sure to complete the classes `Composer` and `Helper` in order for `compose` to behave as expected.

```
class Composer<X, Y, Z> {

    Func<_____, _____> compose(Func<X, Y> f, Func<Y, Z> g) {
          1      2

        ____________________________________________________________________________________;
                                            3
    }

    private class Helper implements Func<_____, _____> {
                                          4      5
        Func<X, Y> f;
        Func<Y, Z> g;

        Helper(Func<X, Y> f, Func<Y, Z> g) {

            ____________________________________________________________________________________;
                                            6

            ____________________________________________________________________________________;
                                            7
        }

        @Override

        ____________________ apply(_____________________________________) {
                  8                          9

            ____________________________________________________________________________________;
                                            10
        }
    }
}
```

6 Lost in Pointers

(16 Points)

Vanessa has implemented a circular `DLList` class, and has created a circular `DLList` (as seen on the reference sheet). However, Dylan the Hacker has gotten into her system and changed one of the `next` or `prev` pointers to point to another node in the `DLList` (or the sentinel). Write a function `fixDLList` that fixes the changed pointer in the instance of the DLList to be the original correct `DLList`.

You may assume that this function is defined in `DLList`, and therefore has access to the private variables of `DLList`.

```
public void fixDLList() {
    int count = 0;
    for (Node n = _____1_____; _____2_____ && n != sentinel; _____3_____) {

        _____________________________________4_____________________________________;
    }

    if (count == size) { // The wrong pointer is a .prev pointer
        sentinel.next.prev = _____________________________________5_____________________________________;

        for (_____6_____; _____7_____; _____8_____) {

            _____9_____ = _____10_____;
        }
    } else { // The wrong pointer is a .next pointer
        sentinel.prev.next = _____________________________________11_____________________________________;

        for (_____12_____; _____13_____; _____14_____) {

            _____15_____ = _____16_____;
        }
    }
}
```

Nothing on this page is worth any points.

Bonus Question

(0 Points)

Write the name of a food dish whose ingredients were first domesticated on as many distinct continents as possible. For example, a pizza: the tomato was domesticated in South America, wheat in Asia, and olives/olive oil in Europe. So the "score" for this food is 3.

Feedback

(0 Points)

Leave any feedback, comments, concerns, or more drawings below!