Submit Exam 2 | Gradescope

Read the exam policies. You have until 12:15pm eastern time to finish this exam. There should be a timer in the top right, which you can hide. Write your name below to acknowledge the exam policies (and for free points).

Q2 Short Answer Questions

12 Points

Your program crashes with a runtime error. The error says it's caused by a NullPointerException which occurs at exactly this line: `PennDraw.text(0.5, 0.5, foo.bar());`. What is the issue and how do you fix it?

Suppose you want to write a function `public static void printGrades(...)` to print a sequence of objects of class `Grade`. Unfortunately, half the grades have been stored in an `ArrayList<Grade>` and the other half are in a `LinkedList<Grade>`. Thankfully, we can print both halves without writing two separate functions. Write the complete signature of `printGrades` so that it takes one argument and can be called with either an `ArrayList<Grade>` or a `LinkedList<Grade>`.

We discussed how getters and setters can be used to protect private instance variables. Suppose you have a `Dog` class that has a private `age` instance variable (an `int`) with a public setter method. Describe concretely an example of what you would be able to do with the setter method to protect `age`.

If the variable `arr` has the type `int[][][]`, what type does `arr[0]` have (assuming it doesn't result in an index out of bounds exception)?

Q3 Find the errors

12 Points

Insertion sort for a `List` is slightly simpler than insertion sort for an array, since `Lists` allow us to move the element directly into its correct spot without swapping. However, there are 4 errors in the insertion sort function below. Find them, describe the error, and how to fix it.

```
1  public static void insertionSort(List<String> l) {
2    for (int i = 1; i < l.size(); i++) {
3      int index = i;
4      for (int j = i; j >= 0; j--) {
5        if (l.get(j).compareTo(l.get(j - 1)) < 0) {
6          index = j;
7        }
8      }
9      String value = l.remove(index);
10     l.add(index, value);
11   }
12 }
```

Q4 Short coding question

12 Points

Write a function `combine` that sequences one linked list after another, but first discarding each of their first elements (if they have a first element). The function should return the head of the combined linked list. This function should work even if there is no first element for either of the input linked lists. Use the following `Node` class for your linked lists:

```
public class Node {
  public int data;
  public Node next;

  public Node(int data, Node next) {
    this.data = data;
    this.next = next;
  }
}
```

Here are some examples, written informally (`[]` is `null`, `[1]` is a linked list with one element where the data is 1, and `+` is applying this `combine` function).

```
[] + [] = []
[] + [1] = []
[] + [1, 2] = [2]
[1, 2, 3] + [] = [2, 3]
[1, 2, 3] + [1] = [2, 3]
[1, 2, 3] + [4, 5, 6] = [2, 3, 5, 6]
```

For instance, the second line `[] + [1] = []` corresponds to the example `combine(null, new Node(1, null)) == null`.

```
public static Node combine(Node head1, Node head2) {
  // TODO
}
```

Write the completed `combine` function below:

Q5 Long coding question

22 Points

```
 (\_/)
(='.'=)
(")_(")
```

We want to represent ASCII art, small text-based drawings like the above, using a class called `Canvas`. This class stores a grid of `char`s, which represents the text drawing. In this question, you will complete the 3 unfinished methods of the `Canvas` class. The constructor and `toString` methods are written for you and you don't need to modify them. There is only one instance variable, `symbol`. Do not change its type or add more instance variables.

```
public class Canvas {
  private char[][] symbols;

  // assume Canvases are always initialized with rows and cols >= 1
```

```
  public Canvas(int rows, int cols) {
    symbols = new char[rows][cols];
    for (int i = 0; i < symbols.length; i++) {
      for (int j = 0; j < symbols[i].length; j++) {
        symbols[i][j] = ' ';
      }
    }
  }

  public String toString() {
    String s = "";

    for (int i = 0; i < symbols.length; i++) {
      for (int j = 0; j < symbols[i].length; j++) {
        s += symbols[i][j];
      }
      s += "\n";
    }

    return s;
  }

  public void setAt(int row, int col, char symbol) {
    // TODO
  }

  public void flip() {
    // TODO
  }

  public void resize(int rows, int cols) {
    // TODO
  }
}
```

Write the `setAt` method which allows you to set the symbol at one of the grid elements of the `Canvas`. The coordinates correspond directly to the array indices. e.g. 0, 0 refers to the top left square. If the coordinates supplied are not within the `Canvas` array bounds, nothing should happen.

Example:

```
Canvas c = new Canvas(3, 3);
c.setAt(0, 0, '1');
c.setAt(1, 1, '2');
c.setAt(2, 2, '3');
c.setAt(2, 3, 'X'); // does nothing
System.out.println(c);
```

should print

```
1
 2
  3
```

Write the `setAt` method below:

Write the `flip` method which flips the picture vertically. i.e. the first row becomes the last, the second row becomes the second last, etc.

Example:

```
// same as first example
Canvas c = new Canvas(3, 3);
c.setAt(0, 0, '1');
c.setAt(1, 1, '2');
c.setAt(2, 2, '3');
c.setAt(2, 3, 'X');

// new stuff
c.flip();
System.out.println(c);
```

should print:

```
  3
 2
1
```

Flipping again should result in the original picture.

Write the `flip` method below:

Write the `resize` method which resizes the canvas. Any existing symbols should be kept as long as they fit on the new canvas. Any new space on the canvas should be filled with `' '` (the space character). Like with the constructor, you can assume that the parameters `row` and `col` are always `>= 1`.

Example:

```
// same as first example
Canvas c = new Canvas(3, 3);
c.setAt(0, 0, '1');
c.setAt(1, 1, '2');
c.setAt(2, 2, '3');
c.setAt(2, 3, 'X');

// new stuff
c.resize(2, 2);
System.out.println(c);
```

should print:

```
1
 2
```

Example:

```
// same as first example
Canvas c = new Canvas(3, 3);
c.setAt(0, 0, '1');
c.setAt(1, 1, '2');
c.setAt(2, 2, '3');
c.setAt(2, 3, 'X');

// new stuff
c.resize(1, 1);
c.resize(3, 5);
```

```
c.setAt(2, 4, 'X');
System.out.println(c);
```

should print:

```
1

    X
```

Write the `resize` method below: