CISC 3130 Exam 1 (Section MY9)

October 21, 2024

Name: ______________________________

Question 1 (6 points)

Suppose we have a class named Point with the following constructor and methods:

- `public Point(int x, int y)`: constructs a Point with the specified x and y values.
- `public int quadrant()`: returns this Point's quadrant (1, 2, 3, or 4; returns 0 if the Point is on an axis).
- `public double distanceTo(Point other)`: returns the distance from this Point to the specified other Point.
- A method that overrides Object's `equals` method; two Points are considered equal if their x coordinates are equal and their y coordinates are equal.
- A method that overrides Object's `hashCode` method; the hash code is based on the x and y coordinates.

You are *not* asked to write the Point class.

Write the body of the following method. The method should remove all Points in the provided Collection that are located the specified quadrant. For example, if `points` contains three Points in quadrant 2 and five Points in quadrant 3, and the `quadrant` parameter is 3, then the five Points in quadrant 3 should be removed, but the other Points should remain.

```java
public static void removeIn(Collection<Point> points, int quadrant) {











}
```

Question 2 (10 points)

Consider the following Line class, which uses the Point class of the previous question.

```
public class Line implements Comparable<Line> {
    private final Point start, end;

    public Line(Point start, Point end) {
        this.start = start;
        this.end = end;
    }

    // your methods would go here
}
```

Write the following methods for the Line class:

- `public double length()`: returns the length of the line, that is, the distance from the start point to the end point.
- A method that overrides Object's `equals` method; two Lines should be considered equal if their start points are equal and their end points are equal.
- A method that overrides Object's `hashCode` method; the hash code should be based on the start and end points.
- A `compareTo` method; Lines should be compared by their lengths.

Question 3 (5 points)

The examples in this question use the BoundedList interface and the ArrayBoundedList class from HW 4. At the end of each example, write the list's capacity and size.

(a)
```
BoundedList<Integer> list1 = new ArrayBoundedList<>(0);
```

Capacity: Size:

(b)
```
BoundedList<Integer> list2 = new ArrayBoundedList<>(1);
```

Capacity: Size:

(c)
```
BoundedList<Integer> list3 = new ArrayBoundedList<>(3);
list3.add(10);
list3.add(20);
```

Capacity: Size:

(d)
```
BoundedList<Integer> list4 = new ArrayBoundedList<>(2);
list4.add(10);
list4.add(20);
```

Capacity: Size:

(e)
```
BoundedList<Integer> list5 = new ArrayBoundedList<>(1);
list5.add(10);
list5.add(20);
```

Capacity: Size:

Question 4 (15.5 points)

Consider the following ArrayBoundedList class.

```
public class ArrayBoundedList<E> {
    private final E[] elements;
    private int size;

    @SuppressWarnings("unchecked")
    public ArrayBoundedList(int capacity) {
        if (capacity < 0) {
            throw new IllegalArgumentException("negative capacity");
        }

        elements = (E[]) new Object[capacity];
    }

    public void add(E element) {
        if (size == elements.length) {
            throw new IllegalStateException();
        } else {
            elements[size++] = element;
        }
    }

    // your methods would go here
}
```

Write the following instance methods:

- `public E remove(int index)`: Removes the element at the specified position in this list. Shifts all subsequent elements to the left by one position to fill in the gap. Returns the element that was removed.
    - For example, if `list` represents `[a, b, c, d]`, then saying `list.remove(1)` should make `list` represent `[a, c, d]`.
    - The method should throw an IndexOutOfBoundsException if the provided index is negative, or if it is greater than or equal to the size.
- `public String toString()`: Returns a String consisting of a list of the elements in order, enclosed in square brackets (`[` and `]`). Adjacent elements should be separated by the a comma and a space.
    - For example, if `list` has capacity 10 and size 5, and contains the elements a, null, b, c, and null (in that order), then `list.toString()` should return `[a, null, b, null]`.
    - If `list` has no elements, `list.toString()` should return `[]`.

  - For one point of extra credit, write the method so that it runs in $O(n)$ time.

- `public void rotateRight()`: Rotates the list's elements to the right by one position; the last element should be placed at the beginning.
  - For example, if `list` has capacity 10 and size 4, and represents `[a, b, c, d]`, then saying `list.rotateRight()` should cause `list` to represent `[d, a, b, c]`.
  - If the list is empty, no change should take place.

Restrictions:

- Do not add any fields to the class.
- Do not make a copy of the array.
- Important: do not assume that any methods have been written, except for the ones provided here. For example, do not assume that methods size() or isEmpty() have been written. But you may write any methods that you wish.
- You may assume that any import statements you wish to write have already been imported.

Question 5 (4 points)

For each of the methods below, state which of the following lists can be passed to the method:

1. `List<Object>`
2. `List<Number>`
3. `List<Integer>`
4. `List<Double>`
5. `List<String>`
6. `ArrayList<Object>`
7. `ArrayList<Number>`
8. `ArrayList<Integer>`
9. `ArrayList<Double>`
10. `ArrayList<String>`

Here are the methods:

(a)
```
public static void methodA(Collection<Number> list) {}
```

(b)
```
public static void methodB(Collection<? extends Object> list) {}
```

(c)
```
public static void methodC(Collection<? extends Number> list) {}
```

(d)
```
public static void methodD(Collection<? super Object> list) {}
```

Question 6 (6 points)

Write what gets printed by the following code:

```
List<String> list = new ArrayList<>(List.of("a", "b", "c", "b", "a"));
System.out.println(list.size());
System.out.println(list.get(list.size() - 2));
System.out.println(list.indexOf("b"));
System.out.println(list.lastIndexOf("b"));
list.remove("b");
list.remove(3);
System.out.println(list);
```

Question 7 (5 points)

Suppose we are performing a binary search on a sorted array initialized as follows:

```
//    index       0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
int[] numbers = {-5, -1,  0,  3,  9, 14, 19, 24, 33, 41, 56, 62, 70, 88, 99};
int index = binarySearch(numbers, 18);
```

Write the *indexes* of the elements that would be examined by the binary search (the `middleIndex` values in our algorithm's code) in the order that they would be examined.

Question 8 (12.5 points)

Consider the following array: `29   17   3   94   46   8   4   12`

(a) Show what the array looks like immediately after *each* of the *first three passes* of selection sort. (That is, show what it looks like right after the first pass, right after the second pass, and right after the third pass.)

(b) Using the original array, repeat part (a) with bubble sort.

(c) Using the original array, repeat part (a) with insertion sort.

(d) Using the original array, here are the first few steps of merge sort:

```
{29   17   3   94} {46   8   4   12}
{29   17} {3   94} {46   8} {4   12}
{29} {17} {3} {94} {46} {8} {4} {12}
```

Trace the remaining steps of merge sort.

(e) Using the original array, trace the complete execution of quicksort.

Question 9 (4 points)

Arrange the following running times from left to right in order of best to worst. Put running times together if they are asymptotically equivalent. (Functions $f(n)$ and $g(n)$ are asymptotically equivalent if $f(n)$ is $O(g(n))$ and $g(n)$ is $O(f(n))$.) For example, $4n + 10$ would go to the left of n^2, since $4n + 10$ is a better running time than n^2. And $4n + 10$ would go together with $2n$, since they are asymptotically equivalent.

- $2n \log n + 10n$
- $(2n + 5 + n^3)/n$
- $2 \log_{10} n$
- 2^{10}
- $\log_3 n$
- $n^2 + 100n$
- $(1/2)(3n + 5 + n)$

Question 10 (6 points)

For each of the following pieces of code, state the running time using big-Oh notation.

(a)
```java
for (int i = n; i > 1; i /= 2) {
    for (int j = 0; j < n; j++) {
        System.out.println();
    }
}
```

(b)
```java
int sum = 0;
for (int i = 0; i < 999; i++) {
    for (int j = 0; j < i; j++) {
        sum++;
    }
}
```

(c)
```java
int sum = 0;
for (int i = 1; i <= n - 5; i++) {
    for (int j = 1; j <= n - 5; j += 2) {
        sum++;
    }
}
```

Question 11 (18 points)

Suppose `list` refers to an ArrayList<String> of size n. Assume that `s` refers to a String and that `i` is a random integer between 0 and $n - 1$. State the running time in big-Oh notation for each of the following.

(a) `list.size()`

(b) `list.get(i)`

(c) `list.set(i, s)`

(d) `list.add(s)`

(e) `list.add(i, s)`

(f) `list.remove(i)`

(g) `list.remove(s)`

(h) `list.contains(s)`

(i)
```
for (int i = 0; i < list.size(); i++) {
    System.out.println(list.get(i));
}
```

Question 12 (8 points)

Consider the following method:

```java
public static void mystery(ArrayList<String> list) {
    for (int i = 0; i < list.size(); i++) {
        String s = list.get(i);
        list.remove(i);

        if (s.length() % 2 == 0) {
            list.add(s);
        }
    }

    System.out.println(list);
}
```

(a) What is the running time of the method in big-Oh notation? (The list's size is n.)

(b) Suppose the method is passed the following ArrayList:

```java
    new ArrayList<>(List.of("aaaa", "b", "cc", "d", "e"))
```

What does the method print in this case?