COS 126 Exam 2 Review Part 1
Q. Can you implement a simple abstract data type?

Example (Fall 2016).

Part 1. Implement a data type ColorHSB.java for HSB colors.

```java
public class ColorHSB {
    public ColorHSB(int h, int s, int b) {
        // create a color
    }
    public String toString() {
        // String representation
    }
    public boolean isGrayscale() {
        // is it gray?
    }
    public int distanceSquaredTo(ColorHSB that) {
        // "distance" to that
    }
}
```

Details.

An HSB color is defined by int values hue (0–360), saturation (0–100), and brightness (0–100).

An HSB color is gray if either its saturation or brightness (or both) is 0.

The distance between two HSB colors is given by the formula

\[
\min((h_1 - h_2)^2, (360 - |h_1 - h_2|)^2) + (s_1 - s_2)^2 + (b_1 - b_2)^2
\]

...
First step. Write a skeleton version of the solution **that compiles**.

```java
public class ColorHSB {
    private final int hue;       // the hue (between 0 and 359)
    private final int saturation; // the saturation (between 0 and 100)
    private final int brightness; // the brightness (between 0 and 100)

    // create a new color with specified hue, saturation, and brightness components
    public ColorHSB(int h, int s, int b) {
        hue = h; saturation = s; brightness = b;
    }

    // return string representation in the format (h, s, b)
    public String toString() {
        return "";
    }

    // is the color a shade of gray?
    public boolean isGrayscale() {
        return true;
    }

    // return squared distance between the two colors
    public int distanceSquaredTo(ColorHSB that) {
        return 0;
    }

    public static void main(String[] args) {
    }
}
```
public class ColorHSB
{
    private final int hue;    // the hue (between 0 and 359)
    private final int saturation;  // the saturation (between 0 and 100)
    private final int brightness;   // the brightness (between 0 and 100)

    // create a new color with specified hue, saturation, and brightness components
    public ColorHSB(int h, int s, int b)
    {
        hue = h; saturation = s; brightness = b; }

    // return string representation in the format (h, s, b)
    public String toString()
    {
        return "(" + hue + ", " + saturation + ", " + brightness + ");";
    }

    // is the color a shade of gray?
    public boolean isGrayscale()
    {
        return saturation == 0 || brightness == 0;
    }

    // return squared distance between the two colors
    public int distanceSquaredTo(ColorHSB that)
    {
        int dh1 = this.hue - that.hue;
        int dh2 = 360 - Math.abs(dh1);
        int ds = this.saturation - that.saturation;
        int db = this.brightness - that.brightness;
        return Math.min(dh1*dh1, dh2*dh2) + ds*ds + db*db;
    }

    public static void main(String[] args)
    {
    }
}
Q. Can you implement and use a simple abstract data type?

Example (Fall 2016).

Part 2. Implement a test client main() that

- Takes three integer command-line arguments h, s, and b.
- Reads a list of pre-defined colors from standard input.
- Prints to standard output the input color that is nearest to (h, s, b).

```
% more web.txt
White 0 0 0
Silver 0 0 75
Red 0 100 100
...

% java ColorHSB 25 84 97 < web.txt
Red (0, 100, 100)
```

You will get test files and required output.
First step. Write a skeleton version of the solution with comments.

```java
public static void main(String[] args) {
    // create color specified on the command line
    // create champion color (and corresponding distance and color)

    // read colors from standard input and find closest color
    while (!StdIn.isEmpty()) {
        // read next color from standard input
        // update champion color if closer
    }

    // print champion color to standard output
}
```
public static void main(String[] args)
{
    // create color specified on the command line
    int h0 = Integer.parseInt(args[0]);
    int s0 = Integer.parseInt(args[1]);
    int b0 = Integer.parseInt(args[2]);
    ColorHSB color0 = new ColorHSB(h0, s0, b0);

    // create champion color (and corresponding distance and color)
    String closestName = null;
    int closestDistance = Integer.MAX_VALUE;
    ColorHSB closestColor = null;

    // read colors from standard input and find closest color
    while (!StdIn.isEmpty())
    {
        // read next color from standard input
        String name = StdIn.readString();
        int h = StdIn.readInt();
        int s = StdIn.readInt();
        int b = StdIn.readInt();
        ColorHSB color = new ColorHSB(h, s, b);

        // update champion color if closer
        int distance = color0.distanceSquaredTo(color);
        if (distance < closestDistance) {
            closestDistance = distance;
            closestName = name;
            closestColor = color;
        }
    }

    // print champion color to standard output
    StdOut.println(closestName + " " + closestColor);
}
Written Exam Logistics

The second exam is on Thursday Dec. 13.
- Covers lectures since first written exam (*not* before).
- Prep session (ADTs, performance, algorithms and data structures) next.
- Prep session (theory and combinational circuits) Tuesday Dec. 11.

You don't all fit in this room.
- Pay attention and know where to go.
- Arrive early.
- No calculator/phone/computer/headphones

Advice.
- Review lectures/reading.
- Try an old exam (untimed).
- Try another one (timed).
- Review a few more.
Example question: Performance

Q. Do you know how to estimate resource requirements of your programs?

Ex. (Fall 2014 WE 1 Question 8) Characterize each of the specified quantities with reference to a function of N as linear, quadratic, cubic, logarithmic, or exp.

Memory use called for by

```java
int[][] a = new int[N][N*N];
cubic
```

Time required to execute

```java
int i = N; while (i>0) i/=2;
logarithmic
```

Time required to execute

```java
int i = N; while (i>0) { int[] a = new int[i]; i/=2; }
linear
```

Time required to execute

```java
String x = "hi"; for (int i=0; i < N; i++) x += x;
exponential
```

String length is \(2^{i+1}\)

The order of growth of the running time of a program that runs for 30 seconds when \(N = 100,000\), 1 minute when \(N = 200,000\), and 1 hour when \(N = 12,000,000\).
Example question: Data types

**Q. Do you understand concepts and Java mechanisms for implementing and using ADTs?**

**Ex. (Spring 2013 Question 3)** Indicate which keyword matches the description on each row.

| defines something as being part of the API | class | public | static | void | main | private | this | null | new |
| creates an instance | | | | | | | | | |
| the value for uninitialized reference variables | | | | | | | | | |
| defines something as being **not** part of the API | | | | | | | | | |
| method name called to start a program | | | | | | | | | |
| return type of a method that returns no value | | | | | | | | | |
| belongs to a class (as opposed to its instances) | | | | | | | | | |
| contains definitions of methods and fields | | | | | | | | | |
| refers to the instance upon which the current method or constructor acts | | | | | | | | | |
Example question: Sorting and searching

Q. Do you know basic properties of classic algorithms for sorting and searching?

Ex. (Fall 2014 Question 6) Describe the order of growth of the running time of each specified algorithm/input below on a file of size $N$.

- **Insertion sort for a randomly ordered file**: quadratic ($N^2$)
- **Mergesort for a randomly ordered file**: linearithmic ($N \log N$)
- **Building a BST for a randomly ordered file**: linearithmic ($N \log N$)
- **Insertion sort for a file that is in reverse order**: quadratic ($N^2$)
- **Insertion sort for a file that is already in order**: linear ($N$)
- **Mergesort for a file that is already in order**: linearithmic ($N \log N$)
- **Building a BST for a file that is already in order**: quadratic ($N^2$)
Example question: Sorting and searching

Q. Do you know basic properties of classic algorithms for sorting and searching?

Ex. (Spring 2015 WE1 Q7) Two sorting algorithms, insertion sort and mergesort, will be used to sort the characters **MELTSONW** into alphabetical order, left-to-right.

The following array contents may occur at some point during either, both, or neither of these sorts. Check all that apply.

<table>
<thead>
<tr>
<th></th>
<th>occurs during insertion sort</th>
<th>occurs during merge sort</th>
<th>does not occur during either</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMLTSONW</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SMLTNOW</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>ELMTNOSW</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ELMSTNOW</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>ELMTSNOW</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Example question: Stacks and queues

Q. Do you know basic properties of fundamental data types?

Ex. (MOOC) When is a stack not a stack? Mark all that apply.

| **When the only operations allowed are to insert an item and to remove the most recently inserted item.** | ✔️ |
| **When the only operations allowed are to insert an item and to remove the least recently inserted item.** |  |
| **When the operation of removing an arbitrary item is supported.** | ✔️ |
| **When the maximum size needs to be specified.** | ✔️ |
| **When the order of growth of the time required to insert an item is logarithmic.** | ✔️ |
| **When the space required cannot be bounded by a constant times the size of the stack at all times.** | ✔️ |
Example question: Linked structures

Q. Do you understand how to write code that manipulates linked structures?

Ex. (Spring 2017 Question 3) Give the code needed to exchange the order of the second and third nodes in a circularly linked list.

Fill in each blank with one of the code snippets shown at right.

```java
private class Node {
    private int item;
    private Node next;
}
```

```java
Node x = first.next
first.next = x.next
x.next = first.next.next
first.next.next = x
```
Example question: BSTs

Q. Do you understand basic properties of binary search trees?

Ex. (1990s) Draw the BST that results when the keys

\[
3 \quad 7 \quad 9 \quad 1 \quad 2 \quad 8 \quad 4 \quad 6 \quad 5
\]

are inserted in that order into an initially empty BST.

Note. We do not ask questions like this any more.

TOO DIFFICULT TO GRADE!
Example question: BSTs

Q. Do you understand basic properties of binary search trees?

Ex. (Spring 2012 Question 7) Suppose that a BST has int values between 1 and 1000. For each sequence listed below, indicate whether or not it could occur during a search for 527.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Could occur during a search for 527</th>
<th>Could not occur during a search for 527</th>
</tr>
</thead>
<tbody>
<tr>
<td>527</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>1 500 600 700 527</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>605 256 490 300 527</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>10 860 523 602 525 527</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>10 860 523 602 599 610 527</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Mark your calendar

**Thursday Dec 6:** PROGRAMMING EXAM 2

**Tuesday Dec 11:** Written exam prep (part 2)

**Thursday Dec 13:** WRITTEN EXAM 2