Vocabulary (16 points)
Define the concepts below, clearly explaining how each element of the pair is related or has something in common, as well as what is distinct or different.

a.) Declaration vs. assignment
b.) Allocation vs. initialization
c.) while loop vs. for loop
d.) accessor vs. manipulator
e.) nested class vs. subclass
f.) listener vs. event handler
g.) Array vs. list.
h.) Abstract data type vs. data structure implementation.

Sorting (16 points)
Consider the following unsorted list of numbers: 10, 4, 13, 7, 8, 2, 11, 7, 15, 9.
For each of the items below indicate which of the three main algorithms we studied might encounter the configuration shown in the process of sorting the list. If a particular configuration would not be produced by any of the three algorithms, write “none”. (The three algorithms are insertion sort, selection sort, and merge sort.)

a.) 2, 4, 7, 7, 8, 10, 11, 13, 15, 9
b.) 2, 7, 8, 11; 4, 7, 9, 10, 13, 15
c.) 4, 7, 8, 10, 13, 2, 7, 9, 11, 15
d.) 2, 4, 7, 10, 13, 8, 11, 7, 15, 9

Stacks (12 points)
Consider a program that uses a stack as its internal data structure. In addition to push() and pop(), the system needs a capacity for undo() and redo(). That is, calling undo() after a push() will take the just-pushed item off the stack, but calling redo() will put it back on. Likewise calling undo() after a pop() will replace the just popped item, and redo() will remove it again. Explain in words and/or pseudo-code how you could write a implement the undo/redo functionality using one or two auxiliary stacks in addition to the main stack. Do not write Java code for this question.
Java Language (16 points)

Consider the code below. Simulate the execution of the program, and predict the output.

```java
public class Refs {
    public int[] p;
    public int q;
    public Refs() {
        p = new int[1];
        p[0] = q = 0;
    }
    public static void m1(int[] p, int q) {
       Refs r = new Refs();
r.p = p;
q = 2;
r.m2(r);
System.out.println(r.p[0]);
System.out.println(r.q);
System.out.println(p[0]);
System.out.println(q);
p[0] = 9;
r.q = 6;
m3(r.p,q);
System.out.println(r.p[0]);
System.out.println(r.q);
System.out.println(p[0]);
System.out.println(q);
    }
    public void m2(Refs r) {
System.out.println(r.p[0]);
System.out.println(r.q);
System.out.println(p[0]);
System.out.println(q);
p[0] = 9;
r.q = 6;
m3(r.p,q);
System.out.println(r.p[0]);
System.out.println(r.q);
System.out.println(p[0]);
System.out.println(q);
    }
    public void m3(int[] p, int q) {
Refs r = new Refs();
r.p[0] = p[0];
q = r.q;
r.q = 5;
p[0] = 4;
    }
    public static void main(String args[]) {
Refs r = new Refs();int[] p = {7};
int q = 8;
r.p[0] = p[0];
r.q = q;
m1(r.p,r.q);
System.out.println(r.p[0]);
System.out.println(r.q);
System.out.println(p[0]);
System.out.println(q);
    }
}
```
Bowling (32 points)
The class below is used to tally scores in a game of bowling. Each game, or “string”, consists of ten “frames”, where each frame is scored separately (for the moment, we are ignoring “spares” and “strikes”; see below).

```java
public class BowlingScore {
    FrameScore first;
    FrameScore last;

    public BowlingScore() {
        first = null;
        last = null;
    }

    public void addFrameScore(int score) {
        FrameScore fs = new FrameScore(score);
        if (first == null) {
            first = fs;
            last = fs;
        } else {
            last.next = fs;
            last = fs;
        }
    }

    public int getFrameScore(int idx) {
        FrameScore fs = first;
        for (int i=0; i<idx; i++) {
            fs = fs.next;
        }
        return fs.data;
    }

    public int total() {
        FrameScore fs = first;
        int tot = 0;
        while (fs.next != null) {
            tot += fs.data;
            fs = fs.next;
        }
        return tot;
    }
}
```

a.) A bowler scores 6, 10, and 3 in her first three frames, and the following instructions are executed. What gets printed to the screen?

```java
BowlingScore bs = new BowlingScore();
bs.addFrameScore(6);
bs.addFrameScore(10);
bs.addFrameScore(3);
System.out.println(bs.getFrameScore(0));
System.out.println(bs.getFrameScore(1));
System.out.println(bs.total());
```

b.) Draw a diagram of the BowlingScore object along with associated FrameScore objects in the manner we've done in class.

c.) The next instruction, shown below, causes a NullPointerException to be thrown. Explain why this happens, and identify the precise line of code where the exception occurs.

```java
System.out.println(bs.getFrameScore(3));
```
d.) Explain how you could modify the code above to gracefully handle the call above (that is, prevent the exception from being thrown).

e.) The `getFrameScore()` function has \( O(n) \), or linear, runtime complexity, where \( n \) is the number of frames bowled. Explain why. How might you modify class `BowlingScore` and/or `getFrameScore()` to run in \( O(1) \), or constant time?

f.) Imagine that we rewrote the `BowlingScore` class using an array of type `int` for internal storage of scores. Think about (but do not write out) how you would implement the constructor, plus the methods `addFrameScore()`, `getFrameScore()`, and `total()`. What would be the performance differences between the two implementations for each method? (In other words, would the running time change in terms of its big-O notation, and if so, how?)

g.) A bowler gets three balls per frame in candlepin bowling. When a bowler knocks down all ten pins with their first two balls, it is called a “spare”, and the score from the first ball of the next frame gets added into the frame with the spare (it is double-counted). How would you modify the code to accommodate this detail? Be specific.

h.) Write a method that takes an array of `BowlingScore` objects and concatenates them together. Assume scores `bs1` holding \( \{2, 4, 6\} \), `bs2` holding \( \{10, 9, 8, 7\} \), and `bs3` holding \( \{1, 0\} \):

```
public BowlingScore concatScores(BowlingScore[] bsArr) {
    // FILL IN HERE
}
```

```
BowlingScore[] bsArr = new BowlingScore[3];
bsArr[0] = bs1;
bsArr[1] = bs2;
bsArr[2] = bs3;
BowlingScore bsAll = concatScores(bsArr);
// above should produce object holding { 2, 4, 6, 10, 9, 8, 7, 1, 0 }
```

Graphical User Interfaces (8 points)

Describe in detail how user actions can trigger programmatic responses. What parts of the program are involved? What preparation does the program need to make before the interface will be responsive?