

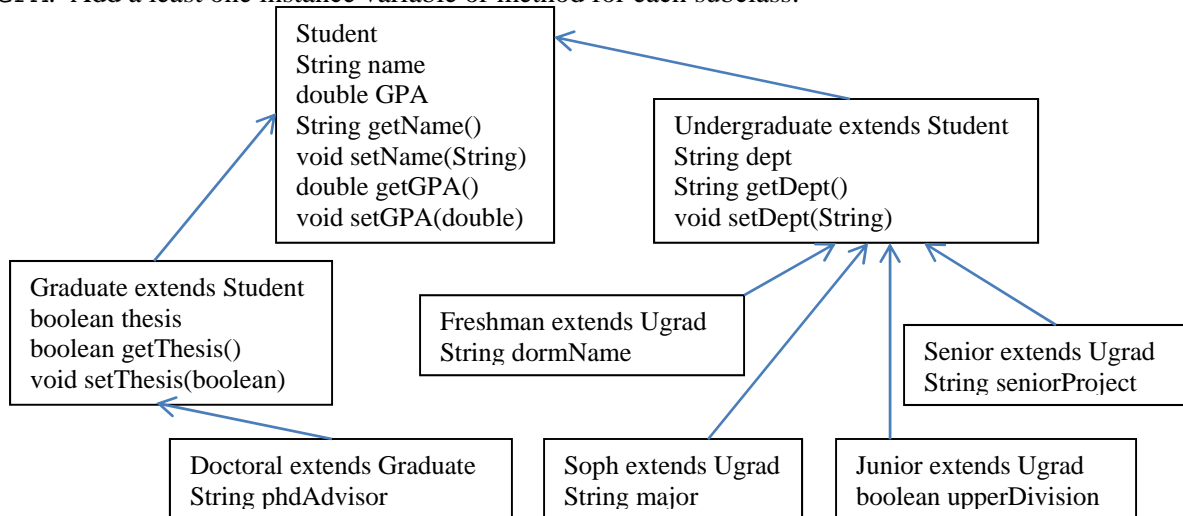
CSCE 145 Sections 1&2: Test 2, Spring 2011

Open book and notes!

1. What are the values of the elements in the array `numbers` after the following code is executed?

```
int[] numbers = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};  
for (int n = 9; n > 0; n = n-3)  
    numbers[n] = numbers[n - 1];  
{10, 20, 30, 30, 50, 60, 60, 80, 90, 90}
```
2. A dorm manager maintains the rooms in a dorm, each of which has a living room (with size in square feet `int livingSF`) and a bedroom (with size in square feet `int bedroomSF`). A room is defined by a class `DormRoom` with two instance variables `livingSF` and `bedroomSF`. Assume there are getters and setters for these variables. Assume the `main` method defines an *array* of many dorm rooms. Write a method called `totalSquareFootage` that accepts an array of `DormRooms` and returns the total amount of square feet in all of the rooms in the array.

```
public int totalSquareFootage(DormRoom[] r) {  
    int sf = 0;  
    for (int n = 0; n < r.length; n++)  
        sf = sf + r[n].getLivingSF() + r[n].getBedroomSF();  
    return sf;  
}
```
3. Write an inheritance hierarchy for classes `Student`, `Undergraduate`, `Graduate`, `Doctoral`, `Freshman`, `Sophomore`, `Junior`, and `Senior`. Use `Student` as the superclass of the hierarchy. Make the hierarchy as deep (i.e., as many levels) as possible. Specify the instance variables and methods for each class. The private instance variables of `Student` should be `name` and `GPA`. Add a least one instance variable or method for each subclass.



4. What results are printed when the *main* method in *TestClass* is executed?

```
public class Flute extends Blue {
    public void method2() {
        System.out.println("Flute 2");
    }
}
public class Blue extends Top {
    public void method1() {
        System.out.println("Blue 1");
    }
}
public class Shoe extends Flute {
    public void method1() {
        System.out.println("Shoe 1");
    }
}
public class Top {
    public void method1() {
        System.out.println("Top 1");
    }
    public void method2() {
        System.out.println("Top 2");
    }
}
public class TestClass {
    public static void main(String[] args) {
        Top[] items = {new Blue(), new Top(), new Shoe(), new Flute()};
        for (int n = 0; n < items.length; n++) {
            item[n].method2();
            item[n].method1();
            System.out.println();
        }
    }
}
```

Top 2
Blue 1

Top 2
Top 1

Flute 2
Shoe 1

Flute 2
Blue 1

5. The following code fragment asks a user over and over again for an index of an array and then prints the value found at that index. Add the **try-throw-catch** code that throws and catches both an `ArrayIndexOutOfBoundsException` and a `NegativeArraySizeException` within the loop.

```
char[] a = new char[50];
int index = 0;
while (true) {
    System.out.println("Enter array index");
    index = keyboard.nextInt();
    try {
        System.out.println("Value is " + a[index]);
    }
}
```

```

    }
}
catch(NegativeArraySizeException e1) {
    System.out.println("Negative index! " + e1.getMessage());
}
catch(ArrayIndexOutOfBoundsException e2) {
    System.out.println("Bad index! " + e2.getMessage());
}
}

```

6. Write a method named *reverseFile* that has one parameter, the name of a text file, and that reads each line of the file and writes it back reversed. The method will not return anything and will not throw any exceptions. For example, if the file **stuff.txt** contained the lines “these are” and “two lines” then after executing *reverseFile*, **stuff.txt** would contain the lines “era eseht” and “senil owt”.

```

public void reverseFile(String fName) {
    String[] contents = new String[100]; // assume 100 lines
    try {
        Scanner in = new Scanner(new File(fName));
        int lineCount = 0;
        while (in.hasNext()) {
            contents[lineCount] = in.nextLine();
            lineCount++;
        }
        in.close();
        PrintWriter out = new PrintWriter(new File(fName));
        String revLine = null;
        for (int n = 0; n < lineCount; n++) {
            revline = "";
            for (int k = 0; k < contents[n].length; k++)
                revline = contents[n].charAt(k) + revline;
            out.println(revline);
        }
        out.close();
    }
    catch(FileNotFoundException e) {
        System.out.println(e.getMessage());
    }
}
}

```