Extra practice questions for you try on your own

1. For this question refer to class X

```
public class X
{
     public void fun () { System.out.println("This is fun"); }
}
```

   a) Write the definition for class Y that so that class Y is a child of class X.

   b) Write an overloaded version of method fun that takes an integer as parameter and displays it onscreen.

   c) Write an overridden version of method fun that takes an integer as a parameter, doubles it and then displays the doubled value onscreen.

2. Trace the output of the following program.

JT's note: parts of this question are fairly easy; as parts of it start getting harder…don't let yourself get freaked. This question just involves concepts that you've already seen: inheritance, overloading, overriding, shadowing. My final hints: (1) you should keep in mind that there are two separate objects (an 'A' object and a 'B' object) (2) as you trace the program write or draw out what is happening each step of the way (don't try to hold it all in your head or you'll likely make many mistakes).

```
public class DriverExtraTrace
{
    public static void main(String [] args)
    {
        A a = new A();
        B b = new B();
        System.out.println(a);
        System.out.println(b);
        a.fun();
        b.fun(123);
        System.out.println(a);
        System.out.println(b);
        a.fun(b.z);
        b.fun(a.y);
        System.out.println(a);
        System.out.println(b);
    }
}
```

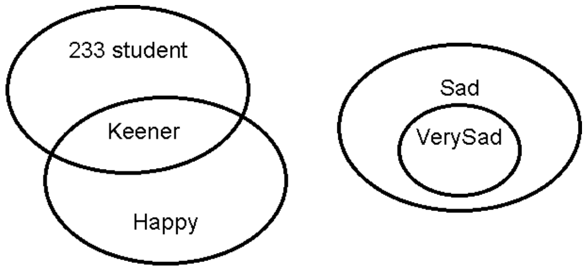
```java
public class A
{
    protected int x;
    protected int y;
    public A() { x = 1; y = 2; }
    protected void fun() { x = x + 1; }
    protected void fun(int a) { y = y + a; }

    public String toString()
    {
        String s = "";
        s = x + " " + y;
        return s;
    }
}

public class B extends A
{
    protected int x;
    protected int z;
    public B() { x = 10; z = 20; }
    protected void fun()
    {
        super.fun(z);
    }
    protected void fun(int a)
    {
        super.fun(a);
    }

    public String toString()
    {
        String s = "";
        s = s + super.x + " " + y + " ";
        s = s + x + " " + z;
        return s;
    }
}
```

3. For this question consider the following Venn diagram. It consists of three sets: 233Student, Happy, Sad and some subsets. If a set (and subset) is implemented as a class definition then draw out the equivalent UML diagram that shows all the sets, subsets as well as their relationships. Can the classes and these relationships be implemented in Java? Why or why not?



<< **Answer** >>

1) What is the output of the following program?

```
public class R4 {
    public static void fun(int num) {
        if (num < 4) {
            num = num + 1;
            fun(num);
        }
        System.out.print(num + " ");
    }

    public static void main(String [] args) {
        fun(1);
    }
}
```

a. 1 2 3 4
b. 4 3 2 1
c. 4 4 3 2
d. 5 4 3 2 1
e. 5 5 4 3 2

Extra-extra practice ideas (I don’t have code solutions but by this point you should be experienced enough to determine – via testing – if your solution is correct). For the questions that involve the use of libraries try to code up the program without looking at exam code, at most you should just refer to a class definition that specifies only method signatures.

- Basic linked list operations: add, remove, display (try writing an iterative and recursive solutions). Create your solution from scratch – don’t have someone else’s code in front of you.
- A very hard linked list operation: reverse order display of a singly linked list
    - Use recursion to display the list in reverse order (last element displayed first, second last element displayed second etc. the first element is displayed last)
    - After displaying the list in reverse order the original ordering of the list should remain unchanged
- Read text data from a file. The program can display different error messages if the file doesn’t exist, if it’s empty, or if there’s some other sort of input output problem. Read a line at a time from the file and display it onscreen (should be able to handle an arbitrary file size).
- Assume with the previous program that the file contains only integers, one on each line. Modify the previous program so each value is converted from a String to an integer and then the number is doubled before being displayed onscreen.
- Write the code that will take the formula for determining your term grade point (see the very first set of notes for this course “admin information”) by calculating a weighted grade point for each component. (JT: although this program is probably easier than what you would face for the final it’s still worth doing because doing extra practice never hurts plus you can now estimate your term grade point to boot!)
- Modify the previous program so that it can handle invalid input (won’t ‘crash’ if the user enters a non-numeric value where a number is expected for a grade).
- Modify the previous program so that it has a graphical rather than a text-based interface. Your choice of GUI controls should be intuitive for the situation e.g., you wouldn’t use a `JTextArea` for just a single line of input. To get the most out of the exercise make sure that the GUI is interactive (so you practice writing code for event handling).
- Another GUI practice idea: implement your first assignment so that it employs a GUI instead of text-based interface.
- Practice your ability to write and trace recursive code by re-writing some of the code that used loops with a recursive solution. (JT: You might want to back up the program beforehand and just modify the copy). If didn’t already do so you might try implementing the linked list ‘search’ operation using recursion rather than a loop.
    - To review the material from before the midterm; try employing some of the operators that you may not have used frequently such as post/pre increment and decrement.

    - To make it extra fun and exciting try doing this with programs that employ concepts from the post midterm part of the course e.g., recursion with pre-post increment.
- Practice idea for code tracing: try writing a program that employs all of the above concepts: overloading, overriding, local variables, variable class attributes, static variables. To get more out of the exercise hand trace the code BEFORE you try running the program. This will let you determine your current level of knowledge and competence. If your hand trace didn't match the actual output then that's a good sign that you have some material to study.