CPSC 233 Final exam review

Short answer 1:

For this question you are to refer to the following Star Trek™ game. The base type of vessel is a 'starship' which has a number of basic attributes and abilities, some of which are shown below:

```
public class StarShip {
	private int hullValue;
	private int shieldValue;

	public int calculateDamage() {
		// Generate and return the damage inflicted on ship
	}

	public void absorbDamage(StarShip attacker) {
		<< Write your answer here >>









		<< End of answer space >>
	}
}
```

The `calculateDamage()` method is used by an attacker to determine the amount of damage that will be inflicted on a defending ship.

The `absorbDamage()` method is used by the defending ship in order to determine the effects of attack damage. This method takes one parameter: a reference to the attacking ship. There are two types of attacking ships: regular starships and JemHadar attack ships. Normally attack damage will be deducted from the shields until they are reduced to zero (shields are 'down') at which point the remaining attack damage is deducted from the hull. (The hull value can potentially reach negative values).

A `JemHadarShip` is a starship that employs a weapon that can match the shield frequency of the defender (anti-proton beam weapon): in effect it can bypass the shields so that damage is always deducted directly from the hull. Only the `JemHadar` ships will have this ability in the game.

Write the code for the `absorbDamage()` method so that: (1) JemHadar starships have the ability to 'penetrate' the shields of the defender during an attack (2) Other attacking starships will first damage the shields and then the hull of the defender. You can only implement this ability by modifying the body of the `absorbDamage()` method. You cannot change the rest of the program nor can you change the signature of this method.

Short answer 2:

For this question you are to modify the based code shown in the program shown below. When the program first runs this program the GUI will look like the image in Figure 1. You are to write the code that will change the text of the button. When the button label says "`On`" it will toggle to "`Off`" when the button is pressed and when the button label says "`Off`" it will toggle to "`On`". You can assume that all the necessary import statements have already been included.

```
public class Driver {
    public static void main(String [] args) {
        JFrame myFrame = new JFrame ("Plain window");
        JButton myButton =new JButton("On");
        myFrame.add(myButton);
        myFrame.setBounds(100,100,300,200);
        myFrame.setVisible(true);
       // Make any changes to the Driver class her









    }
}

public class MyButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
       // Make any changes to the listener class here









    }
}
```

Figure 1

Short answer 3:
What is the output of the following program?

```
public class Driver {
     public static void main(String [] args) {
          List aList = new List();
          aList.display();
     }
}

public class List {
    private Node head;
    public List() {
        final int max = 3;
        int i;
        head = null;
        Node aNode = new Node(0);
        for (i = 0; i < max; i++) {
            aNode.num = i;
            add(aNode);
            aNode.next = null;
        }
    }
    public void add(Node aNode) {
        Node temp;
        if (head == null)
            head = aNode;
        else {
            temp = head;
            while (temp.next != null)
                temp = temp.next;
            temp.next = aNode;
        }
    }

    public void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp);
            temp = temp.next;
        } // while loop
    } // display()
} // Class List
```

```
public class Node {
    public int num;
    public Node next;
    public Node(int aNum) { num = aNum; next = null; }
    public String toString() { return(Integer.toString(num)); }
}
```
<< **Write your answer here** >>

Short answer 4:

Critique the following implementation of the Model-View-Controller design pattern.

```
public class Manager
{
      private Node head = null;
      private char menuSelection;

      public void addElement(Node temp) {
          ...
      }

      public void removeElement(int index) {
          ...
      }

      public void displayList() {
          ...
      }

      public void displayMenu() {
          ...
      }

      public void getMenuSelection() {
          ...
      }
}
```
<< **Write your answer here** >>

Short answer 5:

What is the output of the following program?

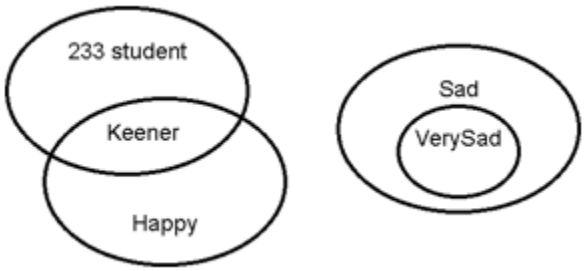
```
public class TraceDriver {
    public static void main (String [] args) {
        A a = new A();
        B b = new B();
        a.fun1();
        a.moreFun();
        b.fun1();
        b.fun2();
        b.fun3();
        b.soFun();
    }
}
public class A {
    public int x;
    public int y;
    public A() { x = 1; y = 2; }
    public void fun1 () {  System.out.println(x + " " + y);  }
    public void moreFun () {  System.out.println("Much fun!");  }
}
public class B extends A {
    public int y;
    public int z;
    public B() { y = 20; z = 30; }

    public void fun1() {  System.out.println(y + " " + z);  }
    public void fun2() {  System.out.println(x + " " + super.x);  }
    public void fun3() {
        super.fun1();
        System.out.println(y + " " + z);
    }
    public void soFun() {
        System.out.println(x + " " + y + " " + z);
    }
}
```

<< **Write your answer here** >>

Short answer 6:

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?



Multiple choice questions:

1) From what you see in the program fragment define below, what will happen when the program reaches the statement inside of class 'Y': `f();`

```
public class X {
       public void f() { ... }
       public void f(int x) { ... }
}

public class Y
{
    public void m() {
        f(); // Question: what happens here?
    }
}
```

a. A call will be made to class X: f()
b. A call will be made to class X: f(int)
c. A call will be made to class Y: m()
d. This will result in a logic error
e. This will result in a syntax error

2) 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).[1]
- Another GUI practice idea: implement your first assignment so that it employs a GUI instead of text-based interface.[1]
- 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.

1 Keep in mind that the final exam is closed book. So when you try writing GUI and file input and output code you shouldn't so with your notes in front of you. Try to do it by referring only the Java API and even then you shouldn't rely on the extra descriptions provided in the API about the attributes and methods. Instead you should be able to write the code by just using a skeleton class definition that only includes attributees and method signatures. Alternatively you can look refer to UML class diagram (which provides the same information).