NAME: ID:

I declare that this is all my own work.

SIGNATURE:

Victoria University of Wellington

Computer Science 205

Midterm Test

April 11, 2005

Answer All Questions
Please Write Neatly
Time Allowed: 50 Minutes
Marks Overall: 50

Numeric Calculators Allowed.
Non-Electronic Translation Dictionaries Allowed.

	Topic	Marks	
1.	Modelling	13 marks	
2.	Designing	12 marks	
3.	Programming	12 marks	
4.	Debugging	13 marks	

1. Modelling [13 Marks]

(a) (4 Marks) Perform a *domain analysis* by textual analysis on the following description of the organisation of a large company, taken from a recent news report.
You should carefully and neatly underline the key nouns in the news report text in the box.

A multinational corporation is divided into subsidiary companies where each subsidiary is quite independent.

Subsidiary companies are directly responsible to the multinational for all of the business in their country. Each subsidiary is divided into a number of regions. Individual regions are in turn made up of individual sales outlets (that is, shops).

The company also has lots of special business units, covering special situations like mail order, the internet, the complaints division. A business unit can belong to a region, to a subsidiary company, or just report directly to the whole multinational.

(b) (2 marks) Based on your textual analysis, list the top 5 *candidate classes* from the news report in the box below.

(c) (7 marks) Draw a UML class diagram showing all the classes and their relationships that you would use to implement the structure of the company.

2. Designing [12 Marks Total]

This code is part of the implementation of a simple program for tracking music played and mixed at a discothèque.

```
public class MusicMix {

 private CdTrack kylie = new CdTrack("Kylie","Locomotion");
 private CdTrack meatloaf = new CdTrack("Meatloaf", "Kick in the HEAD");

 public static void main(String[] args) {
        new MusicMix().dance();
        new MusicMix().getFunky();
 }

 public void dance() { kylie.play(); }

 public void getFunky() {meatloaf.mix(kylie).play(); }

}



public class CdTrack {

 public String artiste;
 public String track;

 public CdTrack(String a, String t) {
         artiste = a; track = t;
 }

 public void play() {
         System.out.println("Now playing " + track + " by " + artiste);
 }

 public CdTrack mix(CdTrack TRACK) {
         return new CdTrack(artiste + " and " + TRACK.artiste, track);
 }

 public void mix(String dj) {
        System.out.println("Now playing " + dj + " remix of " + track);
 }

}
```

This code compiles and runs correctly. Unfortunately, this code has several errors in design and style.

(a) (6 marks)

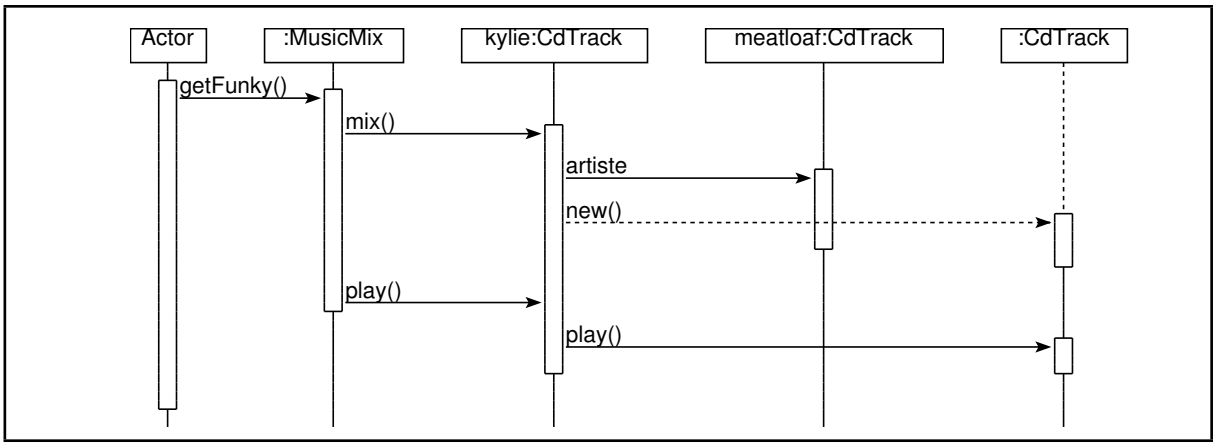
- Circle *two* different design or style errors in the code.
  Do **not** choose errors to do with the lack of comments in the code.
- For each of these two errors:
  Write one line describing the error and and why it is a problem:

i.

ii.

(b) (6 marks) The *Sequence Diagram* shown below is supposed to be the `getFunky()` method in the `MusicMix` code. Unfortunately, it also has several errors.

- Circle *two* of these errors on the sequence diagram.
  Do **not** choose the same errors as you have indicated in the code.
- For each of these two errors:
  Write one line describing the error and why it is a problem:



i.

ii.

3. Programming [12 Marks Total]

A *bag* is a data structure that is just like a set, except that it can contain multiple copies of the same element.

The code on this page and the next page compiles and runs correctly, however it implements a non-generic `BagOfFruit` class that can only store objects of type `Fruit`, and it uses non-generic collection classes.

Carefully and neatly make this class into a generic `Bag` container, so that it can store any kind of objects, using a generic collection class.

Carefully and neatly change the code that uses the `BagOfFruit` class to use your new generic `Bag` class.

```
import java.util.HashMap;
import java.util.Map;

public class BagOfFruit {

    //fruitMap stores the number of each kind of fruit in the bag

    private Map fruitMap = new HashMap();


    /** return how many of this kind of fruit is in the bag

     * @param f - fruit

     * @return how many of that fruit in the bag

     **/
    public Integer howMany (Fruit f) {

        return fruitMap.containsKey(f) ?

            (Integer)fruitMap.get(f) : new Integer(0);
    }


    /** add one more fruit to the bag

     * @param f - the fruit to add

     */
    public void add(Fruit f) {

        fruitMap.put(f, new Integer(howMany(f).intValue() + 1));
    }


    /** remove fruit from bag

     * @param f - fruit to remove

     */
    public void remove(Fruit f) {

        fruitMap.put(f, new Integer(

                    Math.max(howMany(f).intValue() - 1,0)));
    }
}
```

```
class Fruit  {

    private String colour;

    private String name;

    public Fruit(String colour_, String name_) {

        colour = colour_; name = name_;

    }

    public String toString() {return colour + " " + name;}

    public boolean equals(Object other) {

        return toString().equals(other.toString());

   }

    public int hashCode() {

        return toString().hashCode();

    }

}


class Main {

    public static void main(String[] args) {

        BagOfFruit fruitBag = new BagOfFruit();


        fruitBag.add(new Fruit("red", "apple"));

        fruitBag.add(new Fruit("green", "pear"));

        fruitBag.add(new Fruit("green", "pear"));

        fruitBag.remove(new Fruit("green", "pear"));

        fruitBag.add(new Fruit("yellow", "banana"));

        fruitBag.add(new Fruit("green", "pear"));


        // prints 2
        System.out.println(fruitBag.
            howMany(new Fruit("green", "pear")).intValue());


    }
}
```

4. Debugging [13 Marks Total]

Consider the classes on this left-hand page, and show the output of the code on the right-hand page.

```
import java.util.*;

class Sheep implements Iterator<Sheep>, Iterable<Sheep> {

        private int n = 0;

        private Sheep next;

        Sheep() {this(null); System.out.println("pop");}

        Sheep(Sheep n) {next = n; System.out.println("boing!");}

        public String toString() {return "Daisy" + n;}

        public Iterator<Sheep> iterator() {return new Sheep();}

        public boolean hasNext() {System.out.println("baaa " + (++n)); return n<3; }

        public Sheep next() {System.out.println("wool " + (++n)); return next; }

        public void remove() { /* do nothing */ }

        public void follow(Object o) {System.out.println("Sheep follow " + o);}

}


class Goat extends Sheep {

        Goat() {this(null); System.out.println("Say Goat!"); };

        Goat(Sheep n)  {super(n); System.out.println("Say Cheese!");}

        public void follow(Object o) {System.out.println("Goat runs from " + o);}

        public void follow(Sheep s) {System.out.println("Goat attacks " + s);}

        public String toString() {return "Billy";}
}
```

```
Sheep trevor = new Goat();

Sheep cecil = new Sheep(trevor);

cecil.follow("Tourist Tammy");

trevor.follow("Magic Mark");

trevor.follow(cecil);

trevor.follow(trevor);

for (Sheep s : cecil)  {
   if (s.equals(cecil)) {
     System.out.println(
       "Hi Cecil!");
   }
   System.out.println("Bye "+s);
}
```

THIS PAGE LEFT BLANK