TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Student ID: .....

EXAMINATIONS — 2008

TERMS TEST

ENGR 202 / COMP 205

Software Design and Engineering

Time Allowed: 50 minutes

Instructions: There are 50 possible marks on the test.
Answer all questions in the boxes provided.
Every box requires an answer.
If additional space is required you may use a separate answer booklet.
Non-electronic Foreign language dictionaries are allowed.
Calculators ARE NOT ALLOWED.
No reference material is allowed.

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

Question 1. Modelling [12 marks]

Consider the following Java code:

```
public class Politician {
 private String name;
 private Party party;
 public Politician (String n, Party p) {name=n; party=p;}
 public String toString() {return name+"["+party+"]";}
 public Party getParty() {return party;}
}

public class Parliament {
 Collection<Politician> mps = new HashSet<Politician>();
 Politician speaker;
}

public class Committee {
 protected Collection<Politician> members;
 private String name;
 public Committee(String n) {name=n;}
 public void printMembers() {
    for(Politician m : members)  {
          System.out.println(m);}}
 public String getName() {return name;}
}

public class Cabinet extends Committee {
 Cabinet(String n) {super("Cabinet");}
 Collection<Politician> ministers = new LinkedList<Politician>();
}

public class Party {
 String name;
 Party(String n) {n=name;}
}
```

(a) [6 marks] Draw a UML class diagram showing all the classes and their relationships in the Java code.

(b) [3 marks] Draw a CRC card for the Cabinet class.

(c) [3 marks] Explain if this JUnit test passes or fails, and why.

```
@Test public void testEquals() {
     Party lab = new Party("Labour");
     assertEquals(new Politician("Helen",lab), new Politician("Helen",lab));}
```

Student ID: .....

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 2. Programming [12 marks]

The following Java code implements part of the "jPod" Java music player.

```
public class jPod {
  public AbstractList<Track> TRACKS = new LinkedList<Track>();

  /* add Track */
  public void trackAdd(Track t) {TRACKS.add(t);}

  /* remove Track */
  public void trackRemove(int t) {TRACKS.remove(TRACKS.get(t));}

  /* play Track */
  public void play(int i) {
      Display.show(TRACKS.get(i).getName());
      Display.show(TRACKS.get(i).getArtist());
      Player.play(TRACKS.get(i).getMusicData());
  }
}

public class Track {
  private String name;
   private String artist;
    private MusicBytes musicBytes;  //actual bytes of music

   public Track(String n, String a) {name=n; artist=a;}
    public String getName() {return name;}
     public String getArtist() {return artist;}
      public MusicBytes getMusicData() {return musicBytes;}
}
```

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

- Circle and number *four* 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 four errors:
  Write one line in the box below describing the error and and why it is a problem:

(a) [3 marks]

(b) [3 marks]

(c) [3 marks]

(d) [3 marks]

Question 3. Refactoring. [13 marks]

A histogram is a data structure that is like a set, except that it counts the number of times each element has been added. The code on this page and the next page compiles and runs correctly, however it implements a non-generic `Histogram` class that can count any and all kinds of objects, and it uses non-generic collection classes.

(a) [10 marks]

Carefully and neatly make this class into a generic `Histogram` container, so that it can store any kind of objects, using a generic collection class. Then, carefully and neatly change the code that uses the `Histogram` class to use your new generic `Generic` class.

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

public class Histogram {

   //this map counts the number of each element in the histogram
   private Map elementCounts = new HashMap();

   /** return how many of this element are in the histogram
    *@param e - element    */
   public int count(Object e) {

       if (elementCounts.containsKey(e))

           return (Integer) elementCounts.get(e);

       return 0;
   }

   /** add one more element to the histogram
    *@param e - the element to be added     */
   public void add(Object e) {

       elementCounts.put(e, count(e) + 1);
   }

   /** remove element from histogram
    *@param e - element to remove
    *@return true if anyting was removed, else false */

   public boolean remove(Object e) {

       return (elementCounts.remove(e) != null);
   }
}
```

```
public 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 obj) {

       if (this == obj) return true;
       if (obj == null) return false;
       if (!(obj instanceof Fruit)) return false;

       final Fruit other = (Fruit) obj;

       return colour.equals(other.colour) &&

           name.equals(other.name);
   }
   public int hashCode() {return name.hashCode();}
}

public class Main {

   public static void main(String[] args) {

       Histogram fruitHistogram = new Histogram();

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

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

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

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

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

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

       // prints 1
       System.out.println(fruitHistogram.count(new Fruit("green", "pear")));
   }
}
```

(b) [3 marks] The Javadoc for `remove` in `java.util.Collection` says that `remove` "Removes a single instance of the specified element from this collection, if it is present. ... Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call)".

Considering only `remove`, explain whether the `Histogram` class is a strong (behavioural) subtype of `java.util.Collection`.

Student ID: .....

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 4. Debugging [13 marks]

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

```
public class Circle {

  public Circle capo;

  public String copa;

  Circle(String ao, Circle oa) {this(); capo=oa;  copa=ao;}

  Circle() {System.out.println("canti");}

  public String toString() {return copa.toUpperCase();}

  void tutti(Object o) {System.out.println("tutti " + copa + o);
    if (capo!=null) capo.tutti(o);}

  void tutti(Dante d) {System.out.println("bella " + copa + d);
    if (capo!=null) capo.tutti(d);}


  public void reformatzi() {capo.capo=capo; System.out.println("Beatrice!");}
}



public class Dante {

  Dante() {System.out.println(toString());}

  public String toString() {return "Dante";}

}



public class Circlo extends Circle {

  Circlo(String ao, Circle oa) {super(); capo=oa;  copa=ao;}

  Circlo() {System.out.println("canto");}

  void tutti(Object o) {System.out.println("tutto " + copa + o); capo.tutti(o);}

}
```

```
Circle roma = new Circle();
Circle lasciate =
   new Circle("Paradiso",
              null);
roma.copa="Pizza";
```

```
Circle ogne =
 new Circlo("Purgatorio",
            lasciate);
roma.capo=lasciate;
Circle speranza =
 new Circle("Inferno", ogne);
```

```
Dante alighieri = new Dante();
```

```
speranza.tutti(alighieri);
```

```
roma.reformatzi();
```

```
ogne.tutti(alighieri);
```

THIS PAGE LEFT BLANK

****