NAME:	ID:
I declare that this is all my own work.	
SIGNATURE:	

Victoria University of Wellington

COMP205: Software Design and Engineering
Midterm Test

April 28, 2006

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) You have been tasked with designing a program for displaying *maps*. Perform a *domain analysis* by textual analysis on the following description of a *map*.
   You should carefully and neatly underline the key nouns in the description below.

A map represents a geographical region of the globe, such as a continent country or sea.

A region is an area of the map which is either: a water area (for seas and oceans); an urban area (for towns and cities); or, a general land area. Regions may contain other regions (e.g. the North Island is a region of New Zealand containing Wellington).

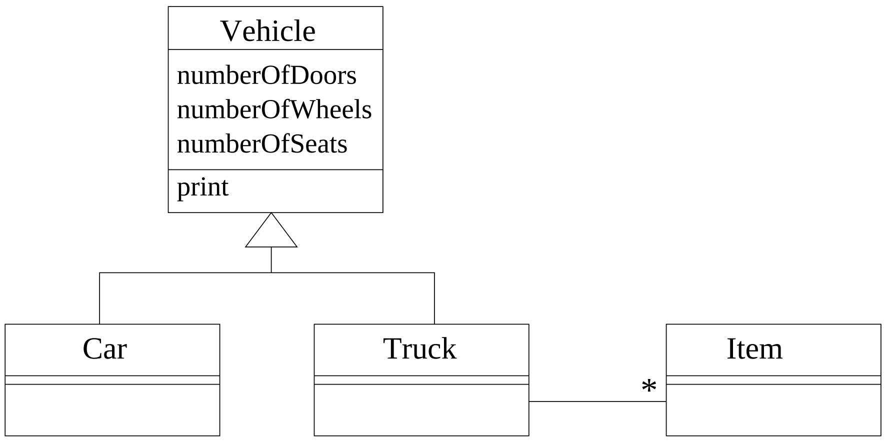
Each region has a name and records its population. Urban areas can be identified as the capital of their country. Land areas may contain certain objects, such as roads, railway tracks, buildings and rivers etc.

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

(c) (7 marks) Draw a UML class diagram showing all the classes and their relationships that you would use to represent a map within the system.

2. Designing [12 Marks Total]

A prototype road vehicle registration system has been designed and built. The UML Class Diagram and its Java implementation are shown below:



```
class Vehicle {
    public int numberOfDoors;
    public int numberOfSeats;
    public int numberOfWheels;
    public Person driver;

    public Vehicle(int nd, int ns, int nw) {
     numberOfDoors=nd;
     numberOfSeats=ns;
     numberOfWheels=nw;
    }
}

public Car {
    private Vehicle sc;  // Super Class
    public Car(int nd, int ns) { super(nd,ns,4,"car"); }
}

public Truck extends Vehicle {
    private Item ITEMS;
    private Truck(int nd, int ns, int nw) { super(nd,ns); }
    private void addItem(Item i) { ITEMS.add(i); }
}
```

Unfortunately, the Java code contains several errors in implementation and style.

(a) (12 marks)

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

i.

ii.

iii.

iv.

v.

vi.

3. Programming [12 Marks Total]

   The Java code shown on this page and the next page contains *six* errors related to Java Generics.

   You must correct each mistake so that the code will compile and run. To do this, remove incorrect code by neatly crossing it out and, as necessary, adding corrected code in the space above each line.

```
class HashRel<FROM,TO> {

 private Map<FROM,Set<TO>> outs = new HashMap<FROM,Set<TO>>();

 private Map<TO,Set<FROM>> ins = new HashMap<TO,Set<TO>>();

 boolean add(FROM f, TO t) {

   from(f).add(t);

   return to(t).add(f);
 }

 Set<Object> from(FROM f) {

  if(outs.get(f) == null) {

   outs.put(f,new HashSet<TO>());

  }

  return outs.get(f);

 }

 Set<FROM> to(TO t) {

  if(ins.get(t) == null) {

   ins.put(t,new HashSet());

  }

  return ins.get(t);

 }

 FROM size() {

  int c=0;

  for(Set<TO> out : outs.values()) {

   c += outs.size();

  }

  return c;
 }

}
```

```
class Attends extends HashRel { // Students can attend upto six Courses
 boolean add(Student s, Course c) {
  if(from(s).size() < 6) {
   return super.add(s,c);
  }
  return false;
 }
}
class Printer { // should print ANY HashRel
 static void printFrom(Object s, HashRel<Object,Object> r) {
  for(Object x : r.from(s)) {
   System.out.println(s + "->" + x);
  }
 }
}
```

4. Debugging [13 Marks Total]

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

```
public class Animal {
 public String n;
 public Animal(String name) { n=name;
  try {
   LittleDog ld = (LittleDog) this; System.out.println(n + " is a LOGD!");
  } catch(Exception e) {
  try {
   BigDog bd = (BigDog) this;
   System.out.println(n + " is a BGDO!");
  } catch(Exception e2) {
   System.out.println(n + " is a TACBL!");
 }}}

 void p(String x) { System.out.println(x); }
 void fight(Animal d) {};
 void fight(LittleDog d) {}; void fight(BigDog d) {};
 void fight(LittleCat c) {}; void fight(BigCat c) {};
}

public class LittleCat extends Animal {
 public LittleCat(String n) { super(n); }
 void fight(Animal a)    { a.fight(this); }
 void fight(BigDog d)    { p(n + " VS " + d.n  + ". " + n + " is EATEN!"); }
 void fight(LittleDog d) { p(n + " VS " + d.n + ". " + n + " is BITTEN!"); }
 void fight(BigCat c)    { p(n + " VS " + c.n + ". " + n + " is SCRATCHED!"); }
 void fight(LittleCat c) { p(n + " VS " + c.n + ". " + n + " is OK!"); }
}

public class BigCat extends Animal {
 public BigCat(String n) { super(n); }
 void fight(Animal a)    { a.fight(this); }
 void fight(BigDog d)    { p(n + " VS " + d.n + ". " + n + " gets SCRATCHED!"); }
 void fight(LittleDog d) { p(n + " VS " + d.n + ". " + d.n + " gets a CLAWING!"); }
 void fight(BigCat c)    { p(n + " VS " + c.n + ". " + n + " is UNSCATHED!"); }
 void fight(LittleCat c) { p(n + " VS " + c.n + ". " + c.n + " was BIT!"); }
}

public class LittleDog extends Animal {
 public LittleDog(String n) { super(n); }
 void fight(Animal a)    { a.fight(this); }
 void fight(BigDog d)    { p(n + " VS " + d.n + ". " + n + " is NOT OK!"); }
 void fight(LittleDog d) { p(n + " VS " + d.n + ". " + d.n + " is OK!"); }
 void fight(BigCat c)    { p(n + " VS " + c.n + ". " + c.n + " CLAWS " + n + "!"); }
 void fight(LittleCat c) { p(n + " VS " + c.n + ". WHAT A SCRAP!"); }
}

public class BigDog extends Animal {
 public BigDog(String n) { super(n); }
 void fight(Animal a)    { a.fight(this); }
 void fight(BigDog d)    { p(n + " VS " + d.n + ". " + n + " SURVIVES!"); }
 void fight(LittleDog d) { p(n + " VS " + d.n + ". " + d.n + " is IN TROUBLE!"); }
 void fight(BigCat c)    { p(n + " VS " + c.n + ". " + n + " BITES " + c.n + "!"); }
 void fight(LittleCat c) { p(n + " VS " + c.n + ". " + c.n + " is NO MORE!"); }
}
```

```
BigCat c1 = new BigCat("Spike");



LittleDog d1 = new LittleDog("Patch");



c1.fight(d1);



Animal d2 = d1;


c1.fight(d2);


Animal d3 = new BigDog("Fang");



Animal c2 = new LittleCat("Teddy");



d3.fight(c2);
```

THIS PAGE LEFT BLANK