NAME:	ID:
SIGNATURE:	

Victoria University of Wellington

Computer Science 205

SAMPLE Test

March 24, 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	12 marks	
2.	Designing	13 marks	
3.	Programming	12 marks	
4.	Debugging	13 marks	

1. Modelling [12 Marks]

   This code is part of the implementation of a simple Banking program.

```
class AuditTrail {
        static void log(String transaction, Account a, int m) {}
        // code not supplied
}

class Account {
        private int balance;
        private int overdraft;

        public Account(int b, int o) { balance = b; overdraft = 0; }

        public int getBalance() { return balance; }
        public int getOverdraft() { return overdraft; }

        public void deposit(int amount) {
            balance += amount;
            AuditTrail.log("deposit", this, amount);
        }
        public void withdraw(int amount) {
            int result = balance - amount;
            if (result < overdraft) { throw new Error("Exceeded Overdraft Limit"); }
            balance = result;
            AuditTrail.log("withdrawl", this, amount);
        }
        public boolean canWithdraw(int amount) {
                return (balance - amount) >= overdraft;
        }


        public boolean transfer(int amount, Account to) {
         if (!canWithdraw(amount)) { return false; }

         withdraw(amount);
         to.deposit(amount);
         return true;
   }
}

class BankingSystem {
        List<Account> accounts = new ArrayList<Account>();


// code here would set up accounts.

        void steal(Account mine) {
                for (Account a : accounts)      {
                        if (!mine.equals(a)) {
                                a.transfer(3, mine);
                        }
                }
        }
}
```

(a) (6 Marks) In the space provided, draw a CRC card that might reasonably have led to the design of the `Account` class.

(b) (6 marks) In the space provided, draw a sequence diagram of the execution of the `BankingSystem`'s `steal` method. You should illustrate the execution of methods that are called by `steal`, choosing which details to show to give a good overview of the most important parts of the method's execution.

2. Designing [13 Marks Total]

Consider the following requirements specification for a very simple aircraft maintenance system:

- The system must keep **Records** for **Aircraft**.
- Each Aircraft has many Records.
- A **Boeing** is a special kind of aircraft. Every Boeing has up to three **SalesRepresentatives**.
- An **Airbus** is a special kind of aircraft. Every Airbus has one **ContactPerson**.
- An aircraft has at least one **Engine**. Engines are part of the aircraft, but can be removed from one aircraft and replaced on the other.
- An aircraft has two **Wings**. Wings are part of the aircraft. If they are broken the whole aircraft must be scrapped; they cannot be removed and attached to another aircraft.

(a) (7 marks) Draw a UML class diagram showing the classes in the aircraft maintenance system.

(b) (6 marks) The *Essential Use Case* card shown below is supposed to be for a use case to record that an aircraft has been repainted. Unfortunately, it has several errors.

Record that an Aircraft has been repainted

User Intention	System Responsibility
	`cout << "Hello";`
Verify ID	
	Type PIN number on terminal keypad
	Swipe card through card reader
Identify repainted aircraft	
	Print Account Balance

- Circle *two* of these errors on the use case card.
- For each of these two errors:
  Write one line describing what it is and why it is a problem:

i.

ii.

3. Programming [12 Marks Total]

Consider this `Set` class and the associated `Student` class.

```
class Set<I> {
  List<I> set_items = new ArrayList<I>();
  int num_items = 0;

 boolean insert(I newitem)
 { if (num_items >= 100) return false;
   set_items.add(num_items,newitem);
   num_items++;
   return true;
 }

 boolean check(I anyitem)
 { int i;
   for (i = 0; i < num_items; i++)
     if ( set_items.get(i) == anyitem)
         return true;
   return false;
 }
}

//-------------------------------------------------
class Student {
  int studentID;
  String surname;
  String initials;
  Student(int id, String sn, String ii) {
        studentID = id; surname = sn; initials = ii; }
  public boolean equals(Object other) {
        if (other instanceof Student) {
                return (((Student)other).studentID == studentID);
        } else return false;
  }
}

//-------------------------------------------------

//
//your declarations would go here
//


students.insert(new Student(40,"Clinton","WJ"));

students.insert(new Student(44,"Bush","GW"));

students.check(new Student(40,"Clinton","WJ"));
```

(a) (3 marks) Write the code to declare a variable called `students` which instantiates the `Set` to store `Students`.

(b) (3 marks) Write the code to declare a variable called `objects` which instantiates the `Set` to store any kind of objects.

(c) (3 marks) Write the code to declare a variable called `integers` which instantiates the `Set` to store integers.

(d) (3 marks) Although you have declared a variable that uses a Set to store Students, the code at the end of the example — calling `check` to verify that student W.J. Clinton, (student number 40) is a member of the `students` set — will return `false`.

Describe why.

4. Class Usage Tracing [13 Marks Total]

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

```
interface Food {
          public String toString();
}

class Animal {
  String name;
  public Animal() {name = "new Cat()"; System.out.println("hare");};
  public Animal(String n) {name = n; System.out.println(this + "wammo");}
  public String toString() {return name;}
  public void eat(Food f) { System.out.println(this + " eats " + f); }
  public void munch(Food f) { this.eat(f); }
}

class Cat extends Animal {
 public Cat(String n) {super(n); System.out.println("" + this + this); name = n;}
 public void eat(Mouse m) {System.out.println(this + " yummy yummy! " + m); }
}

class Mouse extends Animal implements Food {
 public Mouse(String n) { System.out.println("cheep " + n); }
 public void eat(Food f) { System.out.print(this); System.out.print(" eats");
                                        System.out.print(f);}
}

class Chocolate implements Food {
 public String toString() { return "Whittakers's Finest"; }
}
```

```
Animal cat = new Cat("Jerry");
```

```
Cat mouse = new Cat("Tom");
```

```
Mouse confused =
 new Mouse("Cadbury");
```

```
cat.eat(new Chocolate());
```

```
cat.eat(confused);
```

```
mouse.eat(new Food() {
 public String toString() {
  return "COMP205";}});
```