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

EXAMINATIONS — 2007

MID-YEAR

COMP 205
Software Design and Engineering

Time Allowed: 3 Hours

Instructions: There are 180 possible marks on the exam.
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.

Question	Topic	Marks
1.	Domain Analysis	30
2.	Software Modelling	30
3.	The Java Language	30
4.	Principles of Object-Oriented Programming	30
5.	Practices of Software Engineering I	30
6.	Practices of Software Engineering II	30
	Total	180

Student ID: .....

Question 1. Domain Analysis [30 marks]

Consider the following narrative:

> "Mel visits the GoodTrade website and follows the links to create a new account. She enters her details and provides a password. Mel is then automatically sent an email confirming those details. She replies to this and validates her account.
>
> Later, Mel logs on to the GoodTrade system using her password which is verified automatically. She selects an item being sold by Dave445 and places a bid on it. Since this is higher than the other bids on that item, the system records Mel as the highest bidder.
>
> When the auction closes, Mel is declared the winner. Mel's contact information is automatically sent to Dave445, and he emails her with his bank account number. Once Mel's payment has gone through, Dave445 posts the item to Mel in a package."

(a) [5 marks] For each of the following questions, two statements (labelled A-B) have been provided. In each case, you should indicate the correct answer by circling only one of the two choices.

(i) A) The system should be responsible for verifying Mel's password.
B) The system should not be responsible for verifying Mel's password.

(ii) A) The system should be responsible for ensuring the highest bid wins.
B) The system should not be responsible for ensuring the highest bid wins.

(iii) A) The system should be responsible for initiating contact between customers.
B) The system should not be responsible for initiating contact between customers.

(iv) A) The system should be responsible for transferring funds between customers.
B) The system should not be responsible for transferring funds between customers.

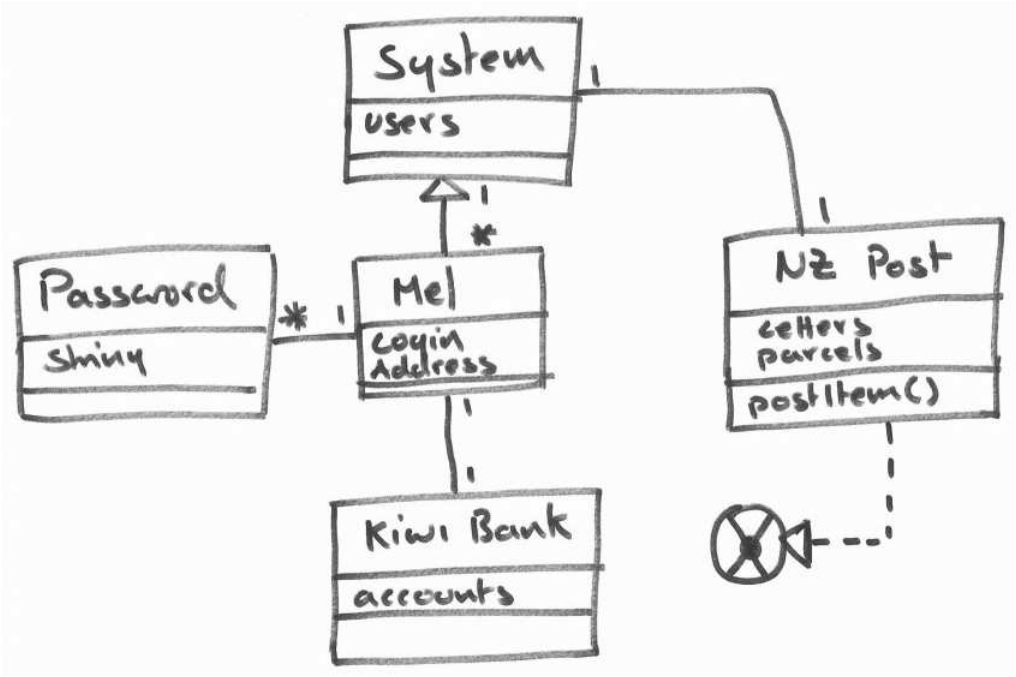
(v) A) The system should be responsible for transporting items between customers.
B) The system should not be responsible for transporting items between customers.

(b) [3 marks] Identify the three most important *candidate use cases* from the narrative.

(c) [12 marks] Draw an *Essential Use Case Card* for each of your three candidate use cases.

Student ID: .....

The following *UML class diagram* represents a partial design for the GoodTrade system:



(d) [10 marks] Circle and number five separate problems with the above UML class diagram. For each problem, write a brief (i.e. one line) description of the problem in the corresponding box below.

1)
2)
3)
4)
5)

Student ID: .....

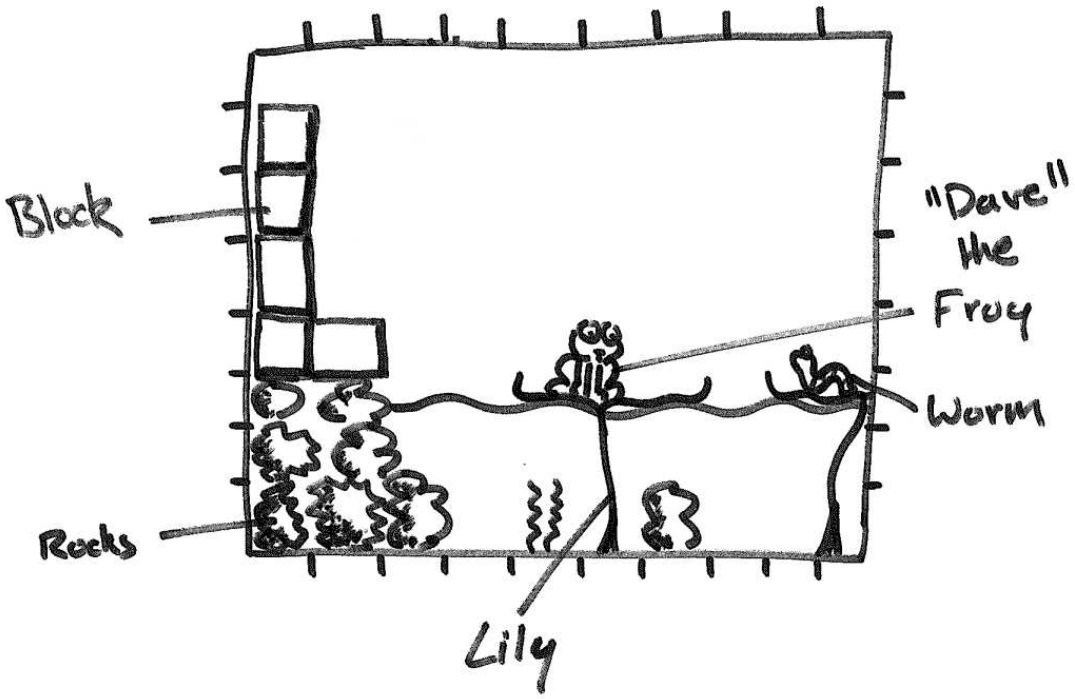
SPARE PAGE FOR EXTRA ANSWERS

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

Student ID: .....

Question 2. Software Modelling [30 marks]

You have been contracted to design software for a simple two-dimensional computer game. A rough description of the game follows:



"In FrogLand there are Frogs of different sizes and colour. Frogs have names and strength levels. Some are strong frogs which can carry things, including rocks, blocks and, of course, other frogs! Frogs are generally peaceful. They like to sit on lilies in the water and eat worms when they are hungry. The hunger level of a frog is measured on a scale of 1-10.

Rocks are found mostly underwater. Concrete blocks provide the foundations for buildings. Rocks and Blocks are solid objects which can be stacked on top of each other. Anything not stacked on a solid object will fall (except frogs which can sit on lilies). Rocks are really the same as blocks — they just have a different graphic to make the world seem more interesting. FrogLand is broken down into a two-dimensional grid of squares, where each item has an x and y coordinate identifying its position."

(a) [10 marks] From the description above, identify ten *candidate objects*.

(b) [15 marks] In the box below, draw a simple *UML Class Diagram* for the FrogLand game. (You should show class attributes in your diagram, but not methods).

(c) [5 marks] In the box below, give the Java code for your class that represents Blocks:

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 3. The Java Language [30 marks]

The following Java code can be used to represent mathematical expressions. It compiles and runs without crashing.

```
public interface Expression extends Cloneable {
 public int evaluate();
 public Object clone();
}

public class Number implements Expression {
 private int num;
 public Number(int n) { num = n; }
 public int evaluate() { return num; }
 public Object clone() { return new Number(num); }
}

public class Add implements Expression {
 protected Expression left, right;
 public Add(Expression l, Expression r) { left = l; right = r; }
 public int evaluate() { return left.evaluate() + right.evaluate(); }
 public Object clone() { return new Add((Expression) left.clone(),
                                        (Expression) right.clone()); }
}

public class Sub extends Add {
 public Sub(Expression l, Expression r) { super(l,r); }
 public int evaluate() { return left.evaluate() - right.evaluate(); }
 public Object clone() { return new Sub((Expression) left.clone(),
                                        (Expression) right.clone()); }
}

public class Test {
 public static void main(String[] args) {
  int j = 10;
  for(int i=0;i!=10;i++) {
   Expression e = new Add(new Number(i), new Number(j));
   System.out.print(e.evaluate() + " ");
   j = j - 2;
  }
 }
}
```

Student ID: .....

(a) (i) [3 marks] In the box below, give the output produced by running the `Test.main()` method.

(ii) [2 marks] In the box below, give Java code for constructing an `Expression` to represent the mathematical statement "$(1 + 2) - 1$".

(iii) [4 marks] Briefly, explain whether the `Sub.evaluate()` method *overloads* or *overrides* the `Add.evaluate()` method.

(iv) [3 marks] The Java code given on the previous page implements a *deep clone*. Briefly, explain what this means.

(v) [5 marks] An instance of `Expression` is not necessarily the same as an `Add` object. Briefly discuss this by considering the ways in which they can and cannot be the same.

(b) Consider the following Java code, which can be used to replace the `Add` and `Sub` classes.

```
public abstract class Op { public abstract int f(int x, int y); }
public class Aop extends Op { public int f(int x, int y) { return x+y; } }
public class Sop extends Op { public int f(int x, int y) { return x-y; } }

public class MultiOp implements Expression {
 private ArrayList<Expression> exprs;
 private Op op;

 public MultiOp(Op o, ArrayList<Expression> es) { op=o; exprs=es; }

 public int evaluate() {
  int r = exprs.get(0).evaluate();
  for(int i=1;i < exprs.size();i++) { r = op.f(r,exprs.get(i).evaluate()); }
  return r;
 }
 public Object clone() {
  return new MultiOp(op, (ArrayList<Expression>) exprs.clone());
}}
```

(i) [3 marks] In the box below, provide Java code which uses the `MultiOp` class to construct an `Expression` representing "$(1+2+3)$".

(ii) [2 marks] Name a *design pattern* used in the `MultiOp` class.

(iii) [4 marks] Discuss whether a `MultiOp` object can be changed after it has been created.

(iv) [4 marks] Does the `MultiOp` class implement a *deep clone*? Justify your answer.

Student ID: .....

Question 4. Principles of Object-Oriented Programming [30 marks]

Consider the following Java code, which compiles and runs without error.

```
public class IntList {
 private List<Integer> ints = new ArrayList<Integer>();
 public int sum = 0;

 public void add(int x) {               // Add an int to list
  sum += x;
  ints.add(new Integer(x));
 }

 public boolean remove(int x) {         // Remove an int from list
  for(int i=0; i < ints.size(); i++) {
   if(ints.get(i) == x) {
    sum -= x;
    ints.remove(i);
    return true;
   }
  }
  return false;
}}
```

(a) [4 marks] For each of the following questions, three statements (labelled A-C) are provided. In each case, indicate the correct answer by circling only one of the three choices.

(i) A) `Integer` is a *subtype* of `int`.
B) `Integer` is a *supertype* of `int`.
C) `Integer` *contains* an `int`.

(ii) A) `List<Integer>` is a *subtype* of `ArrayList<Integer>`.
B) `List<Integer>` is a *supertype* of `ArrayList<Integer>`.
C) `List<Integer>` *contains* an `ArrayList<Integer>`.

(iii) A) Renaming field "`sum`" could break program code outside `IntList`.
B) Renaming field "`sum`" could only break program code inside `IntList`.
C) Renaming field "`sum`" could not break any program code.

(iv) A) Renaming field "`ints`" could break program code outside `IntList`.
B) Renaming field "`ints`" could only break program code inside `IntList`.
C) Renaming field "`ints`" could not break any program code.

(b) [5 marks] Suppose you removed the `sum` field from `IntList` altogether. In the box below, provide a replacement function which computes the sum from scratch.

(c) [5 marks] Suppose you made the `sum` field of `IntList` *private*. Briefly discuss what changes (if any) would be necessary to `IntList` and any code which uses an `IntList`.

(d) [6 marks] Taking into consideration your answers to (b) and (c) above, discuss why *public fields* are considered harmful and how getter/setter methods can help.

(e) Consider the following extension of an `IntList`:

```
public class MulIntList extends IntList {
 public int mul = 1;

 public void add(int x) {
  if(x <= 0) { throw new RuntimeException("x must be > 0!"); }
  mul = mul * x;
  super.add(x);
 }

 public boolean remove(int x) {
  if(x <= 0) { throw new RuntimeException("x must be > 0!"); }
  if(super.remove(x)) {
    mul = mul / x;
    return true;
  }
  return false;
}}
```

(i) [4 marks] State the key idea of the *Liskov Substitution Principle*.

(ii) [6 marks] Explain why `MulIntSet` is not a *strong subtype* of `IntSet`. You may use examples to aid your explanation where necessary.

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 5. Practices of Software Engineering I [30 marks]

Consider the following implementation of a Character Buffer, which compiles without error:

```
public class CharBuffer {
 private char[] buffer;
 private int next = 0;

 public CharBuffer(int max) { buffer = new char[max]; }

 public CharBuffer(char[] charseq) {
  buffer = charseq;
  next = charseq.length;
 }

 public void append(char c) {
  if(next == buffer.length) {
   // not enough space in buffer!
   char[] nbuffer = new char[buffer.length * 2];
   // copy elements from old buffer to new buffer
   System.arraycopy(buffer,0,nbuffer,0,buffer.length);
   // activate new buffer
   buffer = nbuffer;
  }
  buffer[next] = c;
  next = next + 1;
 }

 public char charAt(int index) {
  if(index < 0 || index >= next) { throw new IndexOutOfBoundsException(); }
  return buffer[index];
 }

 // Return size of buffer's active portion
 public int length() { return next; }

 // Construct string from active portion of buffer.
 public String toString() { return new String(buffer,0,next); }
}
```

(a) (i) [2 marks] What happens when `append(char)` is called and there are no more spaces in the `buffer` array?

(ii) [2 marks] Write a suitable *pre-condition* for the `charAt(int)` method.

(iii) [2 marks] Under what circumstance is it impossible to call `charAt(int)` on a `CharBuffer` without raising an exception?

(iv) [2 marks] The `charAt(int)` method throws an exception when an error occurs. Rewrite this method to use an `assert` statement instead.

(v) [2 marks] Give the *class invariant* that should be maintained by `CharBuffer`.

(vi) [5 marks] Do the *constructors* of `CharBuffer` initialise the class invariant properly? Justify your answer.

(vii) [5 marks] Do the *methods* of `CharBuffer` maintain the class invariant properly? Justify your answer.

(b) You have been tasked with *white-box testing* the `CharBuffer` class.

(i) [5 marks] Briefly describe "White-box Testing".

(ii) [2 marks] Provide two suitable test cases for the `append(char)` method.

(iii) [3 marks] In the box below, give Java code to run one of your test cases.

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 6. Practices of Software Engineering II [30 marks]

Consider the following Java code for a `Graph` class, which compiles without error:

```
class Graph {
 private EDGE s = null;   // start of doubly linked list of edges

 // This method adds (head,tail) to list of edges
 public boolean add_edge(int head, int tail) {
 // first, search for matching edge
 for(EDGE p = s; p != null; p=p.next) {
 if(p.head == head && p.tail == tail) { return false; }
 }
 s = new EDGE(s,null,head,tail); // no match, so add new edge!
 return true;
 }

 // This method removes (head,tail) from list of edges
 public boolean removeEdge(int HEAD, int tail) {
  // search for matching edge
  EDGE p = s;
  while(p != null) {
   if(p.head == HEAD && p.tail == tail) {
    // found a match, so remove edge by bypassing it!
    if(p.prev == null) { s = p.next; } else { p.prev.next = p.next; }
    if(p.next != null) { p.next.prev = p.prev; }
    return true;
  }}
  return false; // return false
 }

 // This method removes (head,tail) from list of edges
 public boolean isEdge(int h, int t) { return findMatch(h,t) != null; }

 // Helper method which finds match.
 public EDGE findMatch(int head, int tail) {
  // search for matching edge
  for(EDGE e = s; e != null; e=e.next) {
   if(e.head == head && e.tail == tail) { return e; }
  }
  return null; // return null
 }

 private static class EDGE {
  public EDGE next, prev;
  public int head, tail;
  public EDGE(EDGE n, EDGE l, int head, int tail) {
   next = n; prev = l;
   this.head = head; this.tail = tail;
}}}
```

Student ID: .....

(a) [6 marks] Circle and number six separate problems of style with the `Graph` class on the previous page. For each problem, write a brief (i.e. one line) description of the problem in the corresponding box below.

1)
2)
3)
4)
5)
6)

(b) During testing a bug was found in the `removeEdge()` method. The problem is that `removeEdge()` does not terminate in certain situations.

(i) [3 marks] What is the cause of the problem?

(ii) [3 marks] Under what circumstances does the problem occur?

(c) [8 marks] The `Graph` class does not reuse code effectively. In the box below, provide a simplified version of `removeEdge()` which uses `findMatch()`.

(d) [2 marks] The `Graph` class uses its own implementation of a *linked list*. Briefly explain why this is unnecessary and how the code could be further simplified.

(e) [8 marks] Taking into consideration your answers to (b), (c) and (d), discuss how *code reuse* can lead to better programs.

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.

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.

****