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

**EXAMINATIONS — 2005**

END-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. | Object-Oriented Design | 30 |
| 2. | Design Patterns | 30 |
| 3. | Principles of Object-Oriented Programming | 30 |
| 4. | The Java Language | 30 |
| 5. | Practices of Software Engineering | 30 |
| 6. | Class Invariants | 30 |
| | **Total** | 180 |

Student ID: .....

## Question 1. Object-Oriented Design [30 marks]

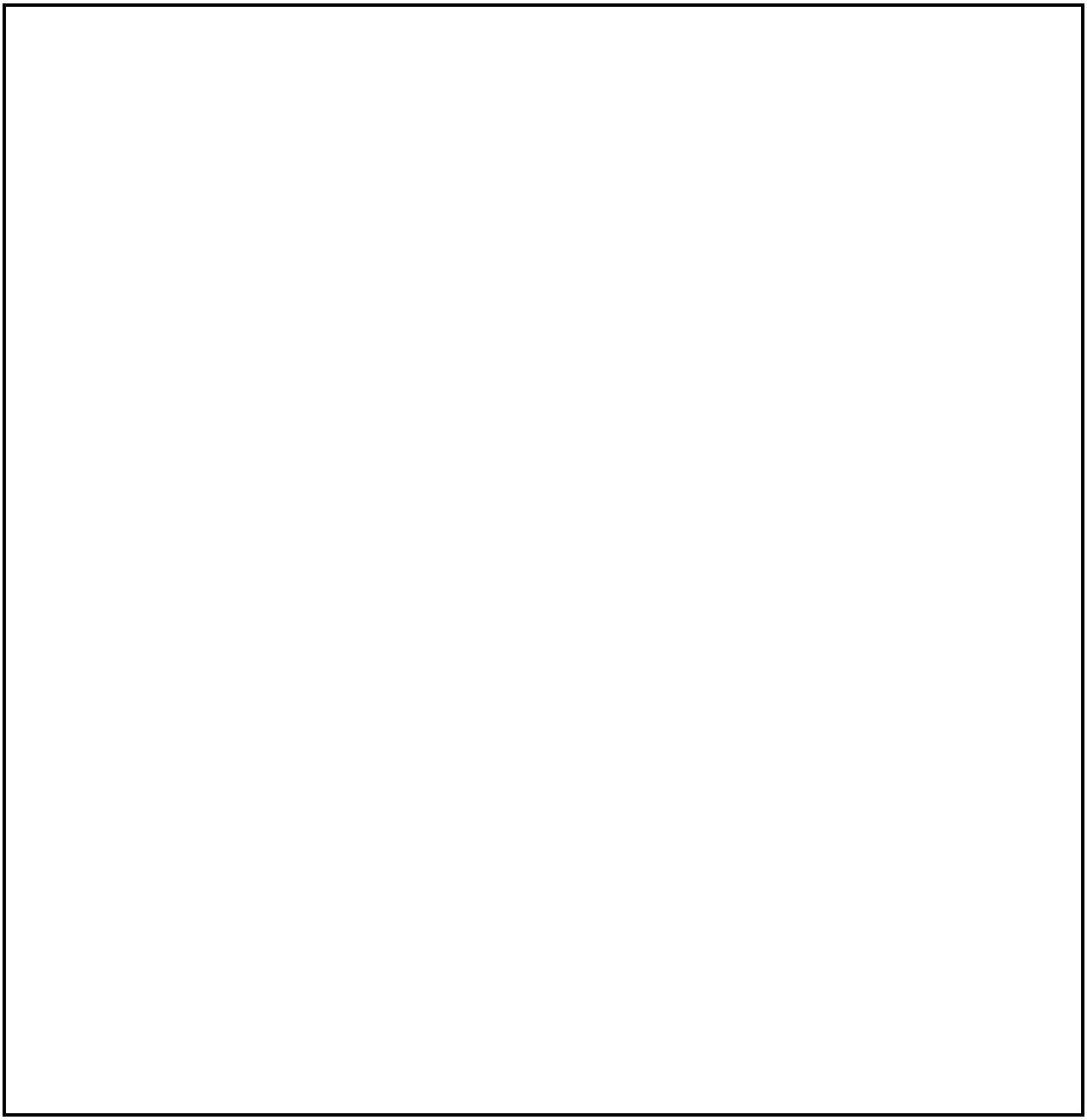
Consider the following narrative:

> Sheila walks up to an automatic airline check-in machine. After watching the animated "Welcome to Palmerston North Airport" introduction screen, Sheila inserts her frequent-flyer card into the machine and types her PIN (personal identity number). The machine looks up the number encoded in Sheila's card in a map, and retrieves the stored Passenger object that represents Sheila.
>
> Sheila then types in her flight code from a display of all flights leaving in the next two hours. The machine then finds the corresponding flight object in a list of all flights leaving today, and asks the flight object to check that Sheila has a reservation for the flight. The flight object allocates a seat on the flight to Sheila, and returns a Seat Booking object representing that Sheila is allocated seat 4A on her flight.
>
> Finally, the machine uses the Seat Booking object to print Sheila's boarding pass, and displays a five-minute animated "Have a nice day!" message.

**(a)** [5 marks] Write an **essential use case** for this narrative.

**(b)** [10 marks] Draw a simple **UML class diagram**, including the following classes, that could be used to implement your essential use case. (You only need to show class names and their relationships: you do not need to show the attributes of each class)

- Check-in Machine
- Map of card number to Passenger
- Passenger
- List of all Flights today
- Flight
- Seat Booking



**(c)** [5 marks] Draw **CRC cards** for the following three classes: *Check-in Machine*, *Flight* and *Seat Booking*.

**(d)** [10 marks] Draw a simple UML sequence diagram illustrating how your essential use case works with the classes in your design.

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. Design Patterns [30 marks]

**(a)** You have been contracted to design software for keeping track of items in a somewhat messy professor's office. A photo of the office follows.

The professor has been in this office for many years and has collected many books, including one he has written (and translations of the book into 4 languages). He also stores conference proceedings and journal publications, some of which are in boxes, and some of which are on his shelf. Sometimes he will put boxes of books or conference proceedings into large boxes and store them under his desk. A few times a year (as dictated by the calender on his wall) students will give him assignments or exams for marking and he will store them on his desk or in a box—hopefully away from his cup of coffee which has been known to spill over unfortunate assignments. Some days he remembers the dangers of storing exams on his desks and puts them in plastic bags. He also has a special shelf for storing bottles of alcohol—the previous occupant of the office left behind a bottle of champagne. He has two other special shelves. One shelf is used to store other shelves — as the shelving is reconfigurable he sometimes takes shelves out and needs to store them somewhere. The other special shelf is used for his electronic equipment, including his computer and speakers.

**(i)** [10 marks] Identify ten items that can be stored in the office, including four containers to put items in or on.

**(ii)** [15 marks] In the box below, draw a class diagram to represent these items.

**(b)** [5 marks] The professor is worried that some of his precious items are being stolen. Using the Decorator pattern, extend your class diagram so that monitoring devices can be added to any item. You should only include classes from your previous diagram which are necessary to describe the extension.

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 3. Principles of Object-Oriented Programming [30 marks]

**(a)** [5 marks] The questions below refer to the following piece of code:

```
abstract class Vehicle {
 void drive(int d);
}

class Car extends Vehicle {
 private int colour;
 void drive(int d) { ... }
}

Vehicle v1 = new Car();
Vehicle v2 = new Car();
v1.drive(1);
```

For each of the following questions, three statements (labelled A-C) have been provided. In each case, indicate which statement is correct by circling only one of the three choices.

**(i)**
A) `Car` is a *super-class* of `Vehicle`.
B) `Car` is a *sub-class* of `Vehicle`.
C) `Car` is *not a class* of `Vehicle`.

**(ii)**
A) `Car` *was a* `Vehicle`.
B) `Car` *has a* `Vehicle`.
C) `Car` *is a* `Vehicle`.

**(iii)**
A) Reflection ensures `v1.drive(1)` calls `Car`'s `drive method`.
B) Polymorphism ensures `v1.drive(1)` calls `Car`'s `drive method`.
C) The statement `v1.drive(1)` does not call `Car`'s `drive method`.

**(iv)**
A) A `Car` object is an instance of the `Car` class.
B) A `Car` class is an instance of the `Car` object.
C) A `Car` class is an instance of the `Vehicle` object.

**(v)**
A) There are many `Car` classes, but there is only one `Car` object.
B) There are many `Car` objects and many `Car` classes.
C) There are many `Car` objects, but there is only one `Car` class.

**(b)** [10 marks] Circle and number five separate problems of style with the following implementation of a Clock class. For each problem, write a brief (i.e. one line) description of the problem in the corresponding box below.

```
public class Clock {
 public int hour;
 public int minute;
 public int SECOND;

 Clock(int x, int min, int sec) {
  // this method initialises clock object
  if(0 <= x && x <= 23 &&
     0 <= min && min <= 59 &&
     0 <= sec && sec <= 59) {
   hour = x;
   // minute = min
   minute = min;
   // second = sec
   SECOND = sec;
  } else { throw new IllegalArgumentException(); }
 }

 void tick() {
  // this method initialises clock object
  if(SECOND != 59) { SECOND++; }
  else if(minute != 59) { SECOND=0; minute++; }
  else if(hour != 23) { SECOND=0; minute=0; hour++; }
}}
```

1.

2.

3.

4.

5.

**(c)** The `protected` keyword is commonly found in Object-Oriented Programming Languages, such as Java and C++.

**(i)** [5 marks] What does this keyword do?

**(ii)** [7 marks] Briefly discuss the advantages and disadvantages of using this keyword in your classes.

**(iii)** [3 marks] Why is there no advantage in having `protected` fields in a `final` class?

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 4. The Java Language [30 marks]

**(a)** Consider the following implementation of a fixed-length vector class:

```
1.  class FixedVector {
2.   public  static int count;
3.   private Object data[];
4.   private int length = 0;
5.
6.   public FixedVector(int size) {
7.    count++;
8.    data = new Object[size];
9.   }
10.
11.  public void add(Object ob) {
12.   if(length < data.length) { data[length++] = ob; }
13.   else { throw new InsufficientSpaceException(); }
14.  }
15.
16.  public Object get(int index) {
17.   if(index < length) { return data[index]; }
18.   else { throw new IndexOutOfBoundsException(); }
19.  }
20.
21.  public Object clone() {
22.   FixedVector c = new FixedVector(data.length);
23.   for(int i=0;i<length;++i) {
24.    c.data[i] = data[i];
25.   }
26.   return c;
27.  }
28.
29.  public static void main(String argv[]) {
30.   FixedVector n = new FixedVector(10);
31.   try {
32.    n.get(1);
33.   } catch(InsufficientSpaceException ie) {
34.    System.out.println("ERROR 1");
35.   } catch(IndexOutOfBoundsException oe)  {
36.    System.out.println("ERROR 2");
37. }}}
```

**(i)** [1 mark] In the implementation of `FixedVector`, how does the `add()` method signal an error has occurred?

**(ii)** [1 mark] Is anything printed when the `main` method is executed? If so, what?

**(iii)** [2 marks] What is being counted by the `count` field?

**(iv)** [2 marks] What would happen if the `count` field was not `static`?

**(v)** [2 marks] There are two standard ways of implementing a `clone()` method: *shallow copy* and *deep copy*. Which is used in `FixedVector`?

**(vi)** [2 marks] Rewrite one line of `FixedVector` so that it uses the other type of clone.

**(vii)** [5 marks] The `FixedVector` class does not use Java generics. Briefly discuss the pros and cons of making `FixedVector` use generics.

**(b)** In UML notation, it is possible to have an *association* between classes. For each of the following examples, briefly describe an appropriate Java implementation.

**(i)** [2 marks]


**(ii)** [3 marks]


**(c)** [5 marks] In Java, it is often possible to use an *interface* instead of an *abstract class*. Briefly describe which of these two mechanisms should be generally preferred and why. You should mention the circumstances in which it makes sense to use the other.

**(d)** [5 marks] Java supports *reflection*. What does this mean?

Student ID: .....

## Question 5. Practices of Software Engineering [30 marks]

**(a)** [5 marks] Briefly describe "Black-Box Testing".

**(b)** [5 marks] You have been assigned the job of testing an Account class. The documentation supplied for this class looks similar to the following:

### class Account

extends `java.lang.Object`

*Represents a bank account allowing deposits and withdrawals.*

| Method Summary | |
|---|---|
| void deposit(Integer) | Deposits an amount into the account. The amount cannot be negative or an exception will be thrown. |

...

Suggest four good test cases for the `deposit()` method.

**(c)** This question deals with the design of an Adventure Game world, such as the game your group designed in the COMP205 project.

An adventure game world is made up of *rooms*. Each room contains a set of *objects*, including *books*, *keys*, *candles* and more. Each object has a long and a short description. The player needs to be able to explore a room by listing the objects it contains and examining their descriptions. The player must also be able to pick up some objects, but not others (e.g. furniture cannot be picked up).

You may draw diagrams to explain your answers to any question, if you feel they are helpful.

**(i)** [12 marks] Briefly describe a simple design for implementing rooms which hold objects. To do this, list the main classes involved and, for each, explain its purpose and justify any uses of inheritance, interfaces, polymorphism or generics. Also describe the main methods/fields necessary to understanding your design. You do not need to give other implementation details.

**(Question 5c continued)**

**(ii)** [5 marks] Describe how your design allows a player to list the objects in a room and examine their descriptions.

**(iii)** [3 marks] Describe how your design could be extended to allow objects to be moved to different positions within a room.

Student ID: .....

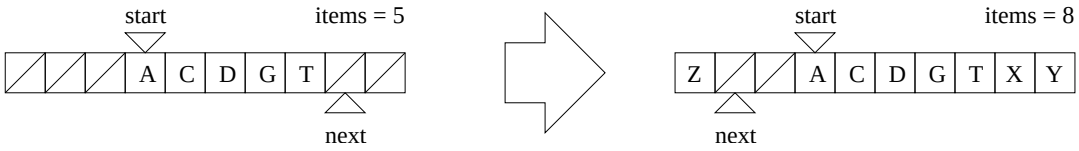
**SPARE PAGE FOR EXTRA ANSWERS**

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

Student ID: .....

## Question 6. Class Invariants [30 marks]

A *cyclic queue* is a FIFO queue implemented using a fixed-size array. Data is pushed on the end and popped from the beginning. The *start* position indicates where to pop the next item from. The *next* position indicates where to push the next item. The following illustrates a cyclic queue before and after three additional items are pushed onto it:




Here, *items* is the number of items on the queue. Notice that *items* equals the number of positions between *start* and *next*. Also, notice that *next* and *start* will "wrap-around" the array when they reach the end.

Consider the following implementation of a *cyclic queue*:

```
class CyclicQueue {
 private Object data[];
 private int items = 0;
 private int start = 0;
 private int next = 0;

 public CyclicQueue(int max) { data = new Object[max]; }

 public void push(Object item) {
  // if queue full, ignore push
  if(items < data.length) {
   data[next++] = item;
   if(next == data.length) { next = 0; }
   items++;
 }}

 public Object pop() {
  Object item = data[start++];
  if(start == data.length) { start = 0; }
  items--;
  return item;
}}
```

**(a)** [2 marks] In the current implementation of `CyclicQueue`, items pushed onto a full queue are ignored. A better approach would be to signal an error has occurred. How could this be done?

**(b)** [2 marks] For an empty queue, no positions in the array are used. In this case, what is the relationship between *start* and *next*?

**(c)** [2 marks] When the queue is full, all positions in the array are used. In this case, what is the relationship between *start* and *next*?

**(d)** [2 marks] Give an example of an invalid state for a `CyclicQueue` object

**(e)** [8 marks] During testing, a bug was found in the implementation of `CyclicQueue`. The bug occurs only when the queue is empty. What is the problem and how could it be fixed?

**(f)** [2 marks] The `push()` and `pop()` methods must preserve the class invariant. What does this mean?

**(g)** [10 marks] Using a simple logic notation, give the class invariant which should be maintained by `CyclicQueue`.

**(h)** [2 marks] Given your above answer, did the original implementations of `push()` and `pop()` preserve the class invariant?

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.

Student ID: . . . . . .

**********