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

EXAMINATIONS — 2010

END-OF-YEAR

SWEN222

Software Design

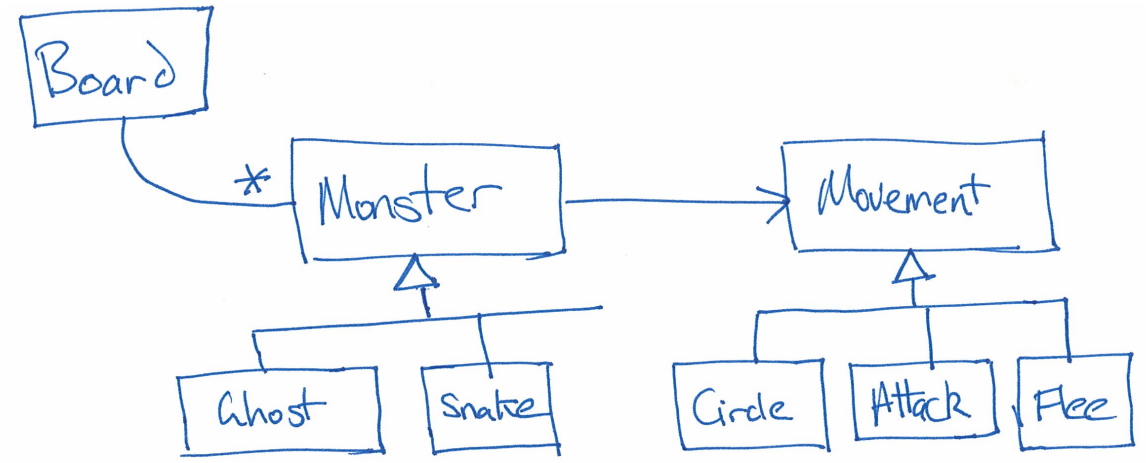
Time Allowed: 2 Hours

Instructions: There are 120 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.	Design Patterns	30
2.	Design By Contract	30
3.	Class Design	30
4.	API Design	30
	Total	120

Student ID: .....

Question 1. Design Patterns [30 marks]



The diagram shows part of the design of a 2-dimensional video game, like PAC-MAN. Every turn, each Monster (Ghost or Snake) has to move to a new square on the board. The `Monster` class decides where to move by asking a `Movement` object to plan its next move:

```
public abstract class Monster {

 // default movement is to move to attack the player
 private Movement movement = new Attack();

 public void move() {
  if (movement != null) {
    Location next = movement.planMove();
    this.animateMoveTo(next);
  }
 }
// ...
}
```

(a) [5 marks] Which pattern is used to plan monsters' movements? Why would this pattern have been chosen here?

Student ID: .....

Consider this specification of the `planMove()` method in the `Movement` interface:

```
interface Movement {
/**
 * Plans the next move for a monster.
 * @return the Location the monster will move to next.
 */
 Location planMove();
}
```

(b) Consider an implementation of `planMove()` that returned a random board `Location`.

(i) [2 marks] Would this method meet the specification?

(ii) [3 marks] Explain why you might want this behaviour in your program.

(c) Consider an implementation of `planMove()` that simply returned the monster's current position.

(i) [2 marks] Would this method meet the specification?

(ii) [3 marks] Explain why you might want this behaviour in your program.

(d) [8 marks] Outline an implementation of the `Attack` class (mentioned on Page 2) which always moves the `Monster` towards the player's current `Location`.

(e) [7 marks] Discuss the design of the `Monster` class and `Movement` interface. Highlight their strengths and weaknesses.

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 By Contract [30 marks]

Consider the following implementation of a linked list, which compiles without error.

```
public class LinkedList<T> {
  private Link head = null;
  private int length = 0;

  public T get(int index) { return find(index).item; }

  public void set(int index, T item) { find(index).item = item; }

  public int length() { return length; }

  public void add(T item) {
    Link<T> ptr = head;
    if(ptr == null) {
      // empty list so append to head
      head = new Link<T>(item);
    } else {
      // non-empty list, so locate last link
      while(ptr.next != null) { ptr = ptr.next; }
      ptr.next = new Link<T>(item);
    }
    length++; // update length cache
  }

  // traverse list looking for link at index
  private Link<T> find(int index) {
    Link<T> ptr = head;
    int i = 0;
    while(i++ != index) { ptr = ptr.next; }
    return ptr;
  }

  private static class Link<S> {
    public S item;
    public Link<S> next;
    public Link(S item) { this.item = item; }
} }
```

(a) (i) [2 marks] What happens if the following piece of code is run?

```
new LinkedList<String>().get(1);
```

(ii) [5 marks] Write suitable *pre-* and *post-conditions* for each of the following methods.

```
int length()
```

```
T get(int index)
```

(iii) [4 marks] Briefly, discuss whether or not a `LinkedList` can be *cyclic* (i.e. contain a link which, either directly or indirectly, refers to itself).

(iv) [5 marks] Give suitable *class invariants* for the `LinkedList` class. You may use written English if you prefer.

(b) Ideas from the *functional programming paradigm* are commonly used to reduce the complexity of object-oriented software. Functional programming emphasises the use of methods which are *side-effect free*.

(i) [3 marks] Briefly, discuss what the term *side-effect free* means.

(ii) [5 marks] State which methods in `LinkedList` are side-effect free, and which are not. For those which are not, briefly state why.

(iii) [6 marks] Briefly, discuss how you could convert `LinkedList` into the functional programming style (i.e. such that all methods are side-effect free).

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. Software Design [30 marks]

(a) This question deals with the design of an Adventure Game world, such as the game your group designed in the SWEN222 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) [15 marks] Briefly describe a simple design for implementing rooms which hold objects. To do this, list the main classes involved, including fields and methods as necessary to understand your design. For each class, briefly explain its purpose and justify any uses of inheritance, interfaces or polymorphism. **You do not need to give other implementation details.**

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

(iii) [5 marks] Describe how your design allows a player to move from one room to the next. If your design does not allow this, describe how it could be extended to do so.

(iv) [5 marks] Describe how your design allows objects to be moved to different positions within a room. If your design does not allow this, describe how it could be extended to do so.

Student ID: .....

Question 4. API Design [30 marks]

Your company is designing the `FunkiPlaya`, a new music application like iTunes but better. This question is about the `FunkiPlaya` interface listed below:

```
/** A FunkiPlaya plays ''funky'' music.  The data can be any kind of Music.
 * A FunkiPlaya has a playlist, and music can be added to the playlist,
 * or removed from it. shuffle() shuffles the current playlist.
 * snapshot() makes a new snapshot. The visualiser can be turned
 * on or off with the funkiVis() method.  Finally, sync() can be used
 * to copy the playlist to an attached iPod or other portable device.
**/
public interface FunkiPlaya {
  /** Adds the Music to the current playlist.
   * @param music - music to add. */
  void getFunki(Music music);

  /** Remove the Music matching name from the current playlist
   * @param Music - name that will be looked up in the playlist
   * @return Music stored with that name, or "MISSING" if not there. */
  Object delete(String name);

  /** Make a new screen snapshot and return it to the user.
   * (Calling snapshot twice writes contents of the playlist to disk)
   * @return a screen snapshot image */
  Object snapshot();

  /** Shuffle the current playlist.
   * @param funki - if TRUE, sort playlist in order of music name.
   *              - if FALSE, do a random shuffle.
   *              - if null, call snapshot() twice and delete the result.
   * @return TRUE if it is a weekend, FALSE otherwise **/
  boolean shuffle(Object funki);

  /** Turn the visualiser on or off.
   * @param i - visualiser pattern to choose.
   * @return true if it worked */
  boolean get(int i);

  /** This syncs all information with an iPod.
   * @return Object array containing all the information. Even numbered
   * entries in this array (0,2,4) have a Music object which is in the
   * playlist. Odd numbered entries are: "DELETED" if that Music object
   * is missing (i.e. has been deleted); "DISK" if that Music is only
   * on disk; or an integer which is the number in the playlist
   * Warning: be very very careful if you update this array!
   * You might break the whole FunkiPlaya! */
   Object[]  dump(); /* sync FunkiPlaya with iPod. May be slow */
}
```

(a) [12 marks] Circle and number six separate problems with the design of the `FunkiPlaya` interface on the previous page. For each problem, write a brief (i.e. one or two line) description of the problem in the corresponding box below.

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.

(b) [18 marks] In the box below, write a Java interface and JavaDoc comments for your own replacement `FunkiPlaya` interface.

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.

****