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

EXAMINATIONS — 2010

MID-YEAR

SWEN221

Software Development

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.	Debugging + Exceptions	30
2.	Inheritance and Polymorphism	30
3.	Testing and Object Contracts	30
4.	Java Generics	30
	Total	120

Student ID: .....

Question 1. Debugging + Exceptions [30 marks]

```
1  public Move parse(String str, boolean isWhite) {
2    int index = 0;
3
4    // first, determine what piece is, and where it's moving from
5    Piece piece = pieceFromStr(str.charAt(index), isWhite);
6    if(!(piece instanceof Pawn)) { index++; }
7    Position start = posFromStr(str.substring(index,index+2));
8    char moveType = str.charAt(index+2);
9    Piece target = null;
10   index = index + 3;
11
12   // second, if this is a take move, determine piece being taken
13   if(moveType == 'x') {
14     target = pieceFromStr(str.charAt(index), !isWhite);
15     if(!(target instanceof Pawn)) { index++; }
16   }
17
18   // third, determine where piece is moving to
19   Position end = posFromStr(str.substring(index,index+2));
20   index = index + 2;
21   Move move;
22   if(target != null) {
23     move = new SinglePieceTake(piece,target,start,end);
24   } else {
25     move = new SinglePieceMove(piece,start,end);
26   }
27
28   // finally, determine if this is a check move
29   if(index < str.length() && str.charAt(index) == '+') {
30     return new Check((MultiPieceMove) move);
31   } else { return new NonCheck((MultiPieceMove) move); }
32 }
33
34 public Piece pieceFromStr(char lookahead, boolean isWhite) {
35   switch(lookahead) {
36     case 'N':
37       return new Knight(isWhite);
38     case 'B':
39       return new Bishop(isWhite);
40     case 'R':
41       return new Rook(isWhite);
42     case 'K':
43       return new King(isWhite);
44     default:
45       return new Pawn(isWhite);
46 } }
```

Student ID: .....

(a) This question concerns the program on the previous page. This program parses Chess moves in long algebraic notation. *You do not need to understand what this is to answer the question.* You may make the following assumptions:

1. Method `posFromStr(String in)` accepts a string of length 2, whose first element is a letter between 'a-h', and the second a digit between '1-8'. If the input string is invalid, an `IllegalArgumentException` is thrown.
2. Method `String.substring(start,end)` returns the characters of a `String` at indexes from `start` up to, but not including, `end`.

For each of the following inputs, state what the `parse()` method returns. If it does not return anything, state what it does instead.

(i) [2 marks] `str == "a4-b5", isWhite == true`

(ii) [2 marks] `str == "b5xKa6", isWhite == false`

(iii) [2 marks] `str == "c4-Kb7", isWhite == true`

(iv) [2 marks] `str == "Rc4pd7", isWhite == true`

(v) [2 marks] `str == "Bc4-d7++", isWhite == true`

(b) [2 marks] The program exhibits an error on input `str=="Qb6-b3",isWhite==true`. It should create a `Queen` object (as Q=Queen). Briefly, describe the problem.

(c) [5 marks] Provide code for `posFromStr()` using the `Position(int row, int col)` constructor, where `row` and `col` are between '1-8'. Here, string `"a1"` should give the position at `row=1, col=1`, whilst `"h2"` gives the position at `row=2, col=8`.

(d) [2 marks] Line 15 of the program on Page 2 contains the statement "`index++`". Briefly, state what this means and how it differs from "`++index`".

(e) [5 marks] Lines 30 and 31 of the program on Page 2 cast instances of `Move` to instances of `MultiPieceMove`. Discuss whether these casts are necessary and, if not, how they can be eliminated.

(f) Consider the following Java code which calls the `parse()` method.

```
1   public Move runParse(String str, boolean isWhite) {
2     Move move = null;
3     try {
4       move = parse(str,isWhite);
5     } catch(IllegalArgumentException e) {
6       Position pos = new Position(1,1);
7       move = new SinglePieceMove(new Pawn(isWhite),pos,pos);
8     } finally {
9       System.out.println("MOVE: " + move);
10    }
11    return move;
12  }
```

(i) [2 marks] Briefly, state what this code does.

(ii) [2 marks] Briefly, discuss whether you think this is an appropriate use of exceptions.

(iii) [2 marks] Briefly, describe how `runParse()` can return **`null`**.

Student ID: .....

Question 2. Inheritance and Polymorphism [30 marks]

(a) Consider the following Java classes,

```
1   interface Strokable { void stroke(); }
2
3   class Animal { int age; }
4
5   class Fish extends Animal {}
6
7   class Cat extends Animal implements Strokable {
8     void stroke() {}
9   }
10
11 class MaineCoon extends Cat {}
```

Given the above declarations, state whether the following code snippets are correct or incorrect. For any which are incorrect, briefly describe the problem.

(i) [2 marks]

```
1   class PetShop {
2     void buy(MainCoon rowan, Animal myPet) {
3         myPet = rowan;
4     }
5   }
```

(ii) [2 marks]

```
1   class FishShop {
2     void buy(Fish fred, MaineCoon myPet) {
3         myPet = fred;
4     }
5   }
```

(iii) [2 marks]

```
1  class StrokingManager {
2    void prioritiseCat(Strokable nextPetToStroke, Cat owen) {
3        nextPetToStroke = owen;
4    }
5  }
```

(iv) [2 marks]

```
1  class Dog extends Animal implements Strokable {
2
3  }
```

(v) [2 marks]

```
1  abstract class FurryPet implements Strokable {
2
3  }
```

(b) Consider the following Java program,

```
1   class Entry {
2     String id = "0";
3     String getId() {
4         return id;
5     }
6     void print() {
7         System.out.println(getId());
8     }
9   }
10  class EntryEight extends Entry {
11    String id2 = "8";
12    String getId() {
13        return id2;
14    }
15
16    void print(String text) {
17        System.out.println(text + "␣" + getId());
18    }
19
20    public static void main(String[] args) {
21        Entry a1 = new Entry();
22        Entry a2 = new EntryEight();
23        a1.print();
24        a2.print();
25    }
26  }
```

(i) [3 marks] What is the output of this program?

(ii) [5 marks] Describe overriding and overloading with reference to the `print` method in the above program.

(c) Consider the following Java code,

```
1  interface Listener { void doTask(); }
2
3  class Widget {
4    private int data;
5    public static class WidgetTask {}
6    public class WidgetHelper{}
7    private Listener getListener() {
8        return new Listener() { void doTask() {} }
9  } }
```

For each of the following classes, provide the following information:

1) The class's name and *kind* (i.e. some or all of normal, static, inner, anonymous).
2) Whether of not instances of the class have a *parent pointer*, and what this means.
3) Example code showing how the class can be instantiated from code outside of the `Widget` class; if this is impossible, then explain why.

(i) [4 marks] `class WidgetTask`:

1)

2)

3)

(ii) [4 marks] `class WidgetHelper`:

1)

2)

3)

(iii) [4 marks] the **`class`** in `getListener`:

1)

2)

3)

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. Testing and Object Contracts [30 marks]

(a) [5 marks] For each of the following groups of statements, clearly indicate the statement which is true:

(i)

1. The JUnit test bar turns green if *at least one* test passes, red otherwise.
2. The JUnit test bar turns green if *most* tests pass, red otherwise.
3. The JUnit test bar turns green if *all* tests pass, red otherwise.

(ii)

1. Code coverage is a measure of how many of your tests cover a part of your program.
2. Code coverage is a measure of how much of a program is covered by your tests.
3. Code coverage is a measure of how much of the original problem your program solves.

(iii)

1. A good unit test tests a single unit of functionality.
2. A good unit test tests exactly six areas of functionality.
3. A good unit test tests as much functionality as possible.

(iv)

1. A white box test tests without knowledge of the implementation.
2. A black box test tests without knowledge of the implementation.
3. A green box test tests without knowledge of the implementation.

(v) Select the correct syntax for performing a JUnit test:

1. `assertArrayEquals("The values are not equal", 102, x)`
2. `assertEqual("The values are not equal", 102, x)`
3. `assert("The values are not equal", 102, x)`

(b) Consider the following Java code, it compiles without error, but is of poor quality,

```
1   class Point {
2     public int x,y;
3
4     public Point(int x, int y) {
5         this.x = x;
6         this.y = y;
7     }
8
9     public boolean equals(Point p) {
10        return this.x == p.x && this.y == p.y;
11    }
12
13    public double distanceFromOrigin() {
14        return Math.sqrt(x*x + y*y)
15    }
16  }
```

(i) [2 marks] Describe a simple way to improve the encapsulation of this class.

(ii) [5 marks] The `equals` method given above is incorrect. The problem is that the method is not always called when we expect it to be. Briefly identify the cause, and give a correct implementation of the `equals` method.

(iii) [2 marks] This class does not work correctly with the `HashMap` class. State how it breaks the required object contract.

(iv) [4 marks] Write a JUnit test case for the `distanceFromOrigin` method.

(v) [6 marks] Write three sensible JUnit tests for the `equals` method. You should include at least one test for an edge case.

Test 1

Test 2

Test 3

(vi) [6 marks] Give code for sorting a collection of `Point` objects. You may not modify the `Point` class. You may find the following method from `java.util.Collections` helpful:

```
static <T> void sort(List<T> list, Comparator<?  super T> c)
```

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. Java Generics [30 marks]

(a) The `CyclicQueue` class, shown below, implements a simple queue holding integer objects.

(i) [6 marks] By writing neatly on the box below, create a generic version of `CyclicQueue`.

```
1  class CyclicQueue {
2
3   private Integer data[] = new Integer[100]; // max 100 items
4
5   private int items = 0, start = 0, next = 0;
6
7   public boolean isEmpty() { return items == 0; }
8
9   public void push(Integer item) {
10
11   if(items < data.length) {
12
13    data[next++] = item;
14
15    if(next == data.length) { next = 0; }
16
17    items++;
18  }}
19
20  public Integer pop() {
21
22   Integer item = data[start++];
23
24   if(start == data.length) { start = 0; }
25
26   items--;
27
28   return item;
29  }}
```

(ii) [4 marks] In the box below, provide code which creates an instance of a generic `CyclicQueue` and puts one item into it.

(iii) [2 marks] State one advantage of having a generic version of `CyclicQueue`, compared with the original.

(iv) [2 marks] Suppose you wanted a generic version of `CyclicQueue` which ensured every object in the queue had a `print()` method. Briefly, state how you would do this.

(v) [4 marks] In the box below, implement a generic method called `emptyList` which accepts a generic `CyclicQueue`, and remove all items until it is empty.

(b) Consider the following Java code for implementing a hierarchy of Shapes:

```
1  interface Shape { }
2  interface ShapeContainer {
3    void add(List<Shape> shapes;}
4    List<Shape> shapes();
5  }
6  public class Square implements Shape { }
7
8  public class SquareHolder implements ShapeContainer {
9    private ArrayList<Square> squares = new ArrayList<Square>();
10
11   public void add(List<Shape> shapes) { squares.addAll(shapes); }
12   public List<Shape> shapes() { return squares;  }
13
14   public String toString() {
15     String r = "";
16     for(String s : squares) { r += s; }
17     return r;
18 } }
```

(i) [9 marks] The above Java code *does not compile correctly*. Identify three errors in the code and, for each, briefly explain the problem.

Error 1

Error 2

Error 3

(ii) [3 marks] Taking your answer(s) to **(i)** into account, comment on the circumstances when, for two generic `Lists` `L1` and `L2`, it is true that `L1` is a subtype of `L2`.

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.

****