Student ID: .....

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

EXAMINATIONS – 2015

TRIMESTER 1

SWEN221

Software Development

Time Allowed: TWO HOURS

CLOSED BOOK

Permitted materials: No calculators permitted.
Non-electronic Foreign language to English dictionaries are allowed.

Instructions: Answer all questions
All questions are of equal value

Answer all questions in the boxes provided.
Every box requires an answer.
If additional space is required you may use a separate answer booklet.

Question	Topic	Marks
1.	Code Comprehension	30
2.	Testing	30
3.	Java Masterclass	30
4.	Exceptions and Assertions	30
	Total	120

Question 1. Code Comprehension [30 marks]

Consider the following classes and interfaces, which compile without error:

```
1  // A variable holding a logic (i.e.  boolean) value
2  class LogicVar {
3    private boolean value;
4
5    public LogicVar(boolean value) { this.value = value; }
6
7    public boolean get() { return value; }
8
9    public void set(boolean value) { this.value = value; }
10 }
11
12 // A logic gate reads two inputs and writes one output
13 abstract class LogicGate {
14   private LogicVar[] variables = new LogicVar[3];
15
16   public LogicGate(LogicVar in1, LogicVar in2, LogicVar out) {
17     variables[0] = in1;
18     variables[1] = in2;
19     variables[2] = out;
20   }
21   public void evaluate() {
22     boolean in1 = variables[0].get();
23     boolean in2 = variables[1].get();
24     variables[2].set(evaluate(in1,in2));
25   }
26   public abstract boolean evaluate(boolean in1, boolean in2);
27 }
28
29 // If both inputs true, out is true; othewise, out is false.
30 class AndGate extends LogicGate {
31   public AndGate(LogicVar v1, LogicVar v2, LogicVar v3) {
32       super(v1,v2,v3);
33   }
34   public boolean evaluate(boolean in1, boolean in2) {
35       return in1 && in2;
36 } }
37
38 // If either input is true, out is true; othewise, out is false.
39 class OrGate extends LogicGate {
40   public OrGate(LogicVar v1, LogicVar v2, LogicVar v3) {
41       super(v1,v2,v3);
42   }
43   public boolean evaluate(boolean in1, boolean in2) {
44       return in1 || in2;
45 } }
```

(a) Based on the code given on page 2, state the output you would expect for each of the following code snippets:

(i) [2 marks]

```
1      LogicVar v1 = new LogicVar(true);
2      System.out.println(v1.get());
```

t

(ii) [2 marks]

```
1  LogicVar v1 = new LogicVar(false);
2  LogicVar v2 = new LogicVar(true);
3  LogicVar v3 = new LogicVar(true);
4  LogicGate gate = new AndGate(v1,v2,v3);
5  gate.evaluate();
6  System.out.println(v1.get() + "␣" + v2.get() + "␣" + v3.get());
```

f t f

(iii) [2 marks]

```
1  LogicVar v1 = new LogicVar(true);
2  LogicVar v2 = new LogicVar(false);
3  LogicGate gate = new OrGate(v1,v2,v2);
4  gate.evaluate();
5  System.out.println(v1.get() + "␣" + v2.get());
```

t t

(iv) [2 marks]

```
1  LogicVar v1 = new LogicVar(true);
2  LogicVar v2 = new LogicVar(false);
3  LogicVar v3 = new LogicVar(false);
4  LogicGate gate1 = new OrGate(v1,v2,v3);
5  LogicGate gate2 = new AndGate(v3,v2,v1);
6  gate1.evaluate();
7  gate2.evaluate();
8  System.out.println(v1.get() + "␣" + v2.get() + "␣" + v3.get());
```

t f f

(b) [5 marks] Provide an implementation for a class `XorGate`. This sets the `out` field to **`true`** if exactly one input is **`true`** (i.e. not both); otherwise, it sets it to **`false`**.

(c) [3 marks] Consider the method `LogicGate.evaluate()`. Does it *overload* or *override* the method `LogicGate.evaluate(`**`boolean,boolean`**`)`? Justify your answer.

(d) Suppose the following method were added to class `LogicGate`:

```
1  public boolean equals(Object o) {
2     if(o instanceof LogicGate) {
3        LogicGate lg = (LogicGate) o;
4        for(int i=0;i!=variables.length;++i) {
5           if(variables[i] != lg.variables[i]) { return false; }
6        }
7        return true;
8     }
9     return false;
10 }
```

(i) [6 marks] This method means an `AndGate` can equal an `OrGate`. Briefly, illustrate how you would fix this problem.

(e) Consider the following snippet of code:

```
1       LogicGate gate = new AndGate(v1,v2,v3);
```

(i) [4 marks] The *static type* of variable `gate` is `LogicGate`. Briefly, discuss what this means and how it affects what values variable `gate` may hold.

(ii) [4 marks] The *dynamic type* of variable `gate` is `AndGate`. Briefly, discuss what this means and how it affects the execution of method `LogicGate.evaluate()`

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 2. Testing [30 marks]

(a) [5 marks] Briefly, discuss the difference between *black-box* and *white-box* testing.

(b) [2 marks] What is *branch coverage*?

(c) [2 marks] What is *simple path coverage*?

(d) Consider the following classes which compiles without error:

```
1  class List {
2    private int[] items;
3
4    public List(int[] items) {
5        this.items = items;
6    }
7
8    public boolean hasBetween(int min, int max) {
9        int i = 0;
10       while(i < items.length) {
11          if(min <= items[i]) {
12             if(items[i] <= max) {
13                 return true;
14             }
15          }
16          i = i + 1;
17       }
18       return false;
19 } }
```

(i) [8 marks] Draw the *control-flow graph* for the `List.hasBetween(int,int)` method:

Consider the following test cases for the class `List`:

```
1 public class ListTests {
2   public static final int[] ITEMS = {-1,0,1};
3
4   @Test public void testHasBetween_1() {
5     assertFalse(new List(ITEMS).hasBetween(5,10));
6   }
7   @Test public void testHasBetween_2() {
8     assertFalse(new List(ITEMS).hasBetween(-10,-5));
9 } }
```

(ii) [2 marks] Give the total *branch coverage* obtained for class `List` from the tests provided in `ListTests`.

(iii) [2 marks] Give the total *simple path coverage* obtained for class `List` from the tests provided in `ListTests`.

(iv) [4 marks] Give two additional test cases which increase the simple path coverage obtained for `List` to 100%.

(e) [5 marks] Briefly, discuss why *polymorphism* in Java can result in an infinite number of execution paths for a given method.

Question 3. Java Masterclass [30 marks]

As for the self assessment tool, for each of the following questions, provide in the answer box the code that should replace `[???]`.

(a) [5 marks]

```
1  //The answer must have balanced parentesis
2  interface Joke{
3   int laughingTime();
4  }
5  class FunnyJoke implements Joke{
6   public int laughingTime(){return 5;}
7  }
8  class BadJoke implements Joke{
9   public int laughingTime(){return 0;}
10 }
11 class SoBadItsGoodJoke extends BadJoke{
12  public int laughingTime(){return 10;}
13 }
14 public class Exercise{
15
16   static int time=0;
17
18   static void joke(Joke j){time+=j.laughingTime();}
19
20   public static void main(String[] arg){
21     joke(new FunnyJoke());
22     joke(new SoBadItsGoodJoke());
23     joke(new BadJoke());
24     assert time==[???];
25   }
26 }
```

(b) [4 marks]

```
1 //The answer must have balanced parentesis
2 class Hero{ int strength(){return 10;} }
3 class [???]{ int strength(){return 100;} }
4 public class Exercise{
5   public static void main(String [] arg){
6     Hero h=new Hercules();
7     assert h.strength()==100;
8   }
9 }
```

(c) [5 marks]

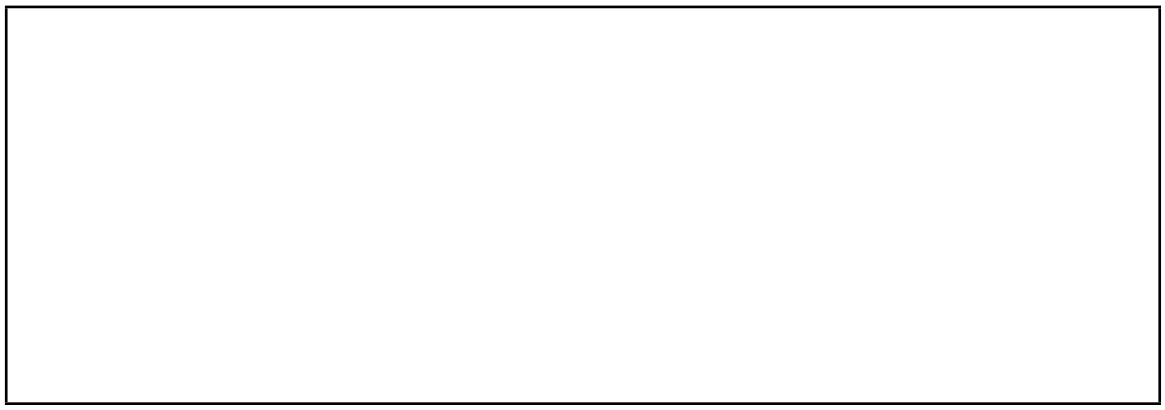
```
1 //The answer must have balanced parenthesis
2 class ThorHammer{[???]}
3
4 public class Exercise{
5   public static void main(String [] arg){
6     ThorHammer h1=ThorHammer.getInstance();
7     ThorHammer h2=ThorHammer.getInstance();
8     assert h1!=null;
9     assert h1==h2;
10 }  }
```

(d) [6 marks]

```
1  //The answer must have balanced parenthesis
2  class Hammer{
3    private int weight;
4    public Hammer(int weight){this.weight=weight;}
5    public int getWeight(){return weight;}
6    public int hashCode() {return this.weight;}
7  }
8  class ThorHammer extends Hammer{[???]}
9
10 public class Exercise{
11   public static void main(String[] arg){
12    assert new ThorHammer().getWeight()==42;
13    assert new Hammer(0).hashCode()==new ThorHammer().hashCode();
14 } }
```

(e) [5 marks]

```
1 //The answer must have balanced parenthesis
2 class A{ int m(){return 1;}}
3
4 public class Exercise{
5   public static void main(String[] arg){
6     A a=[???];
7     assert a.m()==2;
8   }
9 }
```



(f) [5 marks]

```
1  // The answer must have balanced parenthesis
2  import java.util.Arrays;
3  import java.util.List;
4
5  class Point{
6    int x;
7    int y;
8    Point(int x, int y) { this.x=x;this.y=y; }
9  }
10 class ColPoint extends Point {
11   int colour;
12   ColPoint(int x, int y, int colour) {
13     super(x,y);
14     this.colour=colour;
15   }
16 }
17
18 public class Exercise{// make this code compile
19   static void printAll([???]){
20     for(Point p:ps){
21       System.out.println(""+p.x+"␣"+p.y);
22     }
23   }
24   public static void main(String[]arg){
25     List<Point> l1=Arrays.asList(new Point(1,2));
26     List<ColPoint> l2=Arrays.asList(new ColPoint(1,2,0));
27     printAll(l1);
28     printAll(l2);
29   }
30 }
```

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 4. Exceptions and Assertions [30 marks]

(a) [2 marks] Are Assertions in Java enabled or disabled by default?

(b) [2 marks] Explain how to enable/disable assertions either from the command line or from eclipse.

(c) [4 marks] Insert sensible assertions with appropriate error messages into the following code to ensure that the parameter cannot be null and that the result will be positive.

```
public static int distanceFromOrigin(Point p){

    int x=p.x*p.x;

    int y=p.y*p.y;

    int result=x+y

    return result;

}
```

(d) [6 marks] One of your colleagues has written a method `dbQuery`. This method connects to a database, executes a query and returns a list of all the data produced. If there is an error working with the database, `dbQuery` simply propagates a checked exception.

You are using `dbQuery` to write a function to load employers data from a database.

```
1  class LoadData{
2    private static
3    List<Data> dbQuery(String id) throws DBException {
4      /*omitted*/
5    }
6    public static Data load(String id){
7      try{
8        List<Data> data=dbQuery("select␣..."+id);
9        if(data.size()!=1){
10         throw new UncheckedDBException(
11           "Data␣size␣is␣"+data.size());
12         }
13       return data.get(0);
14     }
15     [???]
16   }
17 }
```

As for the self assessment tool, provide in the answer box the code that should replace `[???]` to make the code compile. At this stage, you can assume a class `UncheckedDBException` is declared elsewhere.

(e) [5 marks] Identify an alternative solution for question **(d)** and discuss its pros and cons.

(f) [4 marks] Provide code for the class `UncheckedDBException`, so that the code before could compile.

(g) "Finally" is an important feature of Java exception handling.

(i) [4 marks] Briefly, discuss what **`finally`** means in Java.

(ii) [3 marks] Briefly, describe a situation where using **`finally`** would be sensible.

* * * * * * * * * * * * * * *

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.

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.