Student ID: ....

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA

UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2014

TRIMESTER 1

SWEN221

Software Development

Time Allowed: THREE HOURS

Instructions: Closed Book.
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.

No calculators permitted.
Non-electronic Foreign language dictionaries are allowed.

No reference material is allowed.

Question	Topic	Marks
1.	Debugging and Code Comprehension	30
2.	Java Masterclass	30
3.	Interfaces & Cloning	30
4.	Exceptions	30
5.	Testing	30
6.	Generics	30
	Total	180

Question 1. Debugging and Code Comprehension [30 marks]

Consider the following classes, which compile without error:

```java
// A square on the board
abstract class Square {
   public abstract void attack();
}

// A blank square on the board
public class Blank extends Square {
   public void attack() {}
}

// A monster on the board
public class Monster extends Square {
    private int hitPoints;

    public Monster(int hitPoints) { this.hitPoints = hitPoints; }

    public void attack() { hitPoints --; }

    public boolean isDestroyed() { return hitPoints == 0; }
}

// The board
public class Board {

    // A width * height grid of squares.  Each square
    private Square[][] squares;

    public Board(int width, int height) {
       squares = new Square[width][height];
    }

    public void place(Monster m, int x, int y, int width) {
        for(int i=x;i!=width;++i) {
          squares[i][y] = m;
        }
    }

    public void attack(int x, int y) { squares[x][y].attack(); }
}
```

(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]

```
Board board = new Board(10,10);
Monster m = new Monster(5);
board.place(m,0,0,3);
board.attack(0,0);
System.out.println(m.isDestroyed());
```

(ii) [2 marks]

```
Board board = new Board(10,10);
Monster m = new Monster(2);
board.place(m,0,0,3);
board.attack(1,0);
board.attack(2,0);
System.out.println(m.isDestroyed());
```

(iii) [2 marks]

```
Board board = new Board(10,10);
Monster m = new Monster(2);
board.place(m,0,0,3);
board.place(m,0,5,3);
board.attack(1,0);
board.attack(2,5);
System.out.println(m.isDestroyed());
```

(iv) [2 marks]

```
Board board = new Board(10,10);
Monster m = new Monster(5);
board.place(m,0,0,3);
board.attack(1,5);
System.out.println(m.isDestroyed());
```

(b) [5 marks] Rewrite the `Board` constructor so that it initialises every square in `squares` to a `Blank` square.

(c) [3 marks] Consider the method `Monster.attack()`. Does this *overload* or *override* the method `Square.attack()`? Justify your answer.

(d) [3 marks] The class `Square` is declared as **`abstract`**. Briefly, discuss what this means.

(e) Two squares on the board may refer to the same `Monster`.

(i) [2 marks] Briefly, explain the meaning of this in terms of *objects* and *references*.

(ii) [5 marks] Briefly, discuss what effect this has on how the program works.

(f) [4 marks] Two squares on the board may refer to the same `Blank` square. Briefly, discuss why this does not affect how the program works.

Question 2. 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) [4 marks]

```
//The answer must have balanced parentesis
public class Exercise{
  public static void main(String [] arg){
    int foo=10;
    assert (10==[???]);
    assert (11==[???]);
    assert (12==[???]);
    assert (13==[???]);
  }
}
```

(b) [4 marks]

```
//The answer must have balanced parenthesis,
class Avatar{
  Avatar(String name){this.name=name;}
  String name;
}
class NintendoAvatar extends Avatar{[???]}

public class Exercise{
  public static void main(String [] arg){
    assert (new NintendoAvatar().name.equals("Mario"));
    assert (new NintendoAvatar("Luigi").name.equals("Luigi"));
}  }
```

(c) [4 marks]

```
//The answer must have balanced parenthesis
class Base1{ int m(){return 1;}}
class Base2{ int m(){return 2;}}
class C1 extends Base1{ int m(){[???]}}
class C2 extends Base2{ int m(){[???]}}
public class Exercise{
  public static void main(String [] arg){
    assert new C1().m()==10;
    assert new C2().m()==20;
}  }
```

(d) [4 marks]

```
//The answer must have balanced parentesis
import java.util.HashSet;
class Elem { [???] }
public class Exercise{
  public static void main(String [] arg){
    HashSet<Elem> es=new HashSet<Elem>();
    es.add(new Elem());
    es.add(new Elem());
    es.add(new Elem());
    assert es.size()==1;
}  }
```

(e) [4 marks]

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

(f) [10 marks]

```
//The answer must have balanced parenthesis
import java.util.ArrayList;
interface A{int m();}
public class Test {
  public static void main(String[] arg){
    ArrayList<A> a=new ArrayList<A>();
    for(int i=0;i<10;i++){add(a);}
    assert a.get(0).m()==0;
    assert a.get(1).m()==1;
    assert a.get(7).m()==7;
    assert a.get(9).m()==9;
  }
  [???]
}
```

Hint: since `add()` is called from `main()` but is not declared, you may want to declare it.

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. Interfaces & Cloning

[30 marks]

(a) Consider the following classes and interfaces:

```
interface Shape {
    boolean contains(int x, int y);
    Shape clone();
}

public class Rectangle implements Shape {
  private int x1;
  private int y1;
  private int x2;
  private int y2;

  public Rectangle(int x1, int y1, int x2, int y2) {
    this.x1 = x1; this.y1 = y1;
    this.x2 = x2; this.y2 = y2;
  }
  public bool contains(int x, int y) {
    // Check x,y is contained within this rectangle
    return x >= Math.min(x1,x2) &&
           x <= Math.max(x1,x2) &&
           y >= Math.min(y1,y2) &&
           y <= Math.max(y1,y2);
  }
  public Shape clone() { [???] }
}
```

(i) [3 marks] Give an appropriate implementation of `clone()` for the `Rectangle` class.

(ii) [5 marks] Briefly, discuss why there is no difference between a *deep clone* and a *shallow clone* for the `Rectangle` class.

(b) Consider the following implementation of shape:

```
class ShapeUnion implements Shape {
    private Shape[] shapes;

    public ShapeUnion(Shape[] ss) {
       this.shapes = ss;
    }

    public boolean contains(int x, int y) {
       for(Shape s : shapes) {
           if(s.contains(x,y)) { return true; }
       }
       return false;
    }

    public Shape clone() { [???] }
}
```

(i) [7 marks] Give an implementation of `clone()` for the `ShapeUnion` class which implements a *deep clone*. You may assume that a `Shape` cannot contain itself.

(ii) [5 marks] Suppose that a `Shape` was permitted to contain itself. Briefly, discuss how you would alter your `clone()` method to handle this.

(c) [5 marks] Consider again the constructor for ShapeUnion:

```
public ShapeUnion(Shape[] ss) {
   this.shapes = ss;
}
```

This constructor assigns the ss parameter directly to the shapes field. Briefly, discuss whether you think this is a good or bad idea and what (if anything) you would do differently.

(d) [5 marks] An InverseShape contains a Shape and includes all those points *not* in the contained Shape. Give an implementation for InverseShape.

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 [30 marks]

A New Zealand supermarket chain has a data management system which is able to recover employers data. This is their `getData()` method

```java
public Employer getData(int id){
  Employer result=null;
  DBConnection db=new DBConnection("...");
  Result r=db.query("select␣..."+id+"...");
  // if a result comes back, just return it
  if(r.size()==1){result=new Employer(r);}
  // otherwise, must have been an invalid id.
  return result;
}
```

(a) [2 marks] How does the current implementation handle the case of an invalid employer ID?

(b) [5 marks] How would you modify this method in order to provide a better behaviour in the case of an invalid employer ID? Write down the new code for method `getData()`.

(c) [4 marks] Write down the complete source code of any exception class that you would define for your answer to (b).

(d) [5 marks] Class `DBConnection` offers a method `close()`. The current implementation does not close the database connection. How would you modify this method in order to ensure the database connection is closed?

(e) `Throwable` is the common supertype for all Java exceptions.

(i) [2 marks] Is `Throwable` checked or unchecked?

(ii) [6 marks] Explain why it must be checked / unchecked and why it could not be otherwise. You are encouraged to use a code example.

(iii) [6 marks] The `Exception` class has the following constructor:
`Exception(String message, Throwable cause)`

Describe the meaning of both parameters. In particular, when is the second parameter useful?

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. Testing [30 marks]

(a) Consider the following classes which compile without error:

```
public class Rectangle {
  private int x1;
  private int y1;
  private int x2;
  private int y2;

  public Rectangle(int x1, int y1, int x2, int y2) {
    this.x1 = x1; this.y1 = y1;
    this.x2 = x2; this.y2 = y2;
  }

  public boolean contains(int x, int y) {
    int minX;
    int maxX;
    int minY;
    int maxY;
    // Determine minimum and maximum bounds
    if(x1 < x2) { minX = x1; maxX = x2; }
    else {
      minX = x2; maxX = x1;
    }
    if(y1 < y2) { minY = y1; maxY = y2; }
    else {
      minY = y2; maxY = y1;
    }
    // Check whether point x,y is contained
    if(minX > x) { return false; }
    if(maxX < x) { return false; }
    if(minY > y) { return false; }
    if(maxY < y) { return false; }
    return true;
} }

public class RectangleTests {
  @Test void testContains_1() {
    assertTrue(new Rectangle(0,0,5,5).contains(1,1));
  }
  @Test void testContains_2() {
    assertTrue(new Rectangle(5,5,0,0).contains(1,1));
  }
  @Test void testContains_3() {
    assertFalse(new Rectangle(0,0,5,5).contains(-1,1));
  }
  @Test void testContains_4() {
    assertFalse(new Rectangle(0,0,5,5).contains(6,1));
} }
```

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

(ii) [2 marks] What is *statement coverage*?

(iii) [2 marks] Give the total *statement coverage* of class `Rectangle` obtained from the tests in `RectangleTests`.

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

(v) [2 marks] Give the total *branch coverage* of class `Rectangle` obtained from the tests in `RectangleTests`.

(b) The *path coverage* criterion counts the proportion of all possible execution paths which are tested.

(i) [3 marks] Give the total number of possible execution paths through the method `Rectangle.contains()`.

(ii) [2 marks] Give the total *path coverage* of class `Rectangle` obtained from the tests in `RectangleTests`.

(iii) [4 marks] Give two additional test cases which increase the path coverage obtained for `Rectangle`.

(iv) [2 marks] Briefly, describe what an *infeasible path* is.

(v) [3 marks] Why is path coverage impossible to measure in general?

Question 6. Generics

[30 marks]

Consider the following code

```
import java.util.ArrayList;

class Point{
  int x;  int y;
  Point(int x, int y){ this.x=x;  this.y=y; }
}
class ColoredPoint extends Point{
  int color;
  ColoredPoint(int x, int y,int color) {
    super(x, y); this.color=color;
  }
}
```

(a) [5 marks] There are many possible representations for colours. The class `ColoredPoint` uses an **`int`**. Write instead a generic class `GenericPoint<T>` that uses any kind of type as a representation of a colour.

(b) Consider the following code

```
1  public class GenericTest {
2    static void m(ArrayList<Point> p){
3     [???]//you will be asked to fill the hole here
4    }
5
6    public static void main(String[] args){
7      ArrayList<ColoredPoint> cps=new ArrayList<ColoredPoint>();
8      try{
9        m((ArrayList<Point>)(Object)cps);
10     }
11     catch(Throwable t){}
12     for(ColoredPoint p:cps){
13       System.out.println(p.color);
14     }
15   }
16 }
```

Initially, Bob the programmer tried to pass variable `cps` directly to the method `m()`, but this caused a compilation error; he could not understand the reason for such an error, thus he decided to trick the type system and cast the error away (line 9).

(i) [5 marks] Explain the effect of the two casts in line 9, i.e. what happens when `m((ArrayList<Point>)(Object)cps);` is executed.

(ii) [7 marks] Inserting such casts is unsafe! Provide an example implementation of the method `m()`, (replacing the `[???]` sign) that forces the method `main()` to throw an exception.

(iii) [8 marks] In your own words, explain why an exception was thrown above.

(iv) [5 marks] What Bob really wanted, was to allow the method `m()` to take both `ArrayList<ColoredPoint>` and `ArrayList<Point>`. Write down a suitable generic method signature for method `m()`, so that both the following calls would be accepted:

```
m(new ArrayList<ColoredPoint>());
m(new ArrayList<Point>());
```

but the following would be rejected

```
m(new ArrayList<String>());
```

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

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.