TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2016

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.	Java Generics	30
3.	Java Masterclass	30
4.	Java 8	30
	Total	120

Question 1. Code Comprehension [30 marks]

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

```
interface Pipe {
    void write(int item);
}
```

```
class Buffer implements Pipe {
    private int[] items;
    private int writePos;
    private int readPos;

    public Buffer(int len) { items = new int[len]; }

    public void write(int item) {
        items[writePos] = item;
        writePos = writePos + 1;
    }
    public int read() {
        int item = items[readPos];
        readPos = readPos + 1;
        return item;
    }
}
```

```
class NegativeFilter implements Pipe {
    private Pipe next;

    public NegativeFilter(Pipe n) { next = n; }

    public void write(int item) {
        if (item >= 0) { next.write(item); }
    }
}
```

```
class Accumulator implements Pipe {
    private Pipe next;
    private int sum;

    public Accumulator(Pipe n) { next = n; }

    public void write(int item) {
        sum = sum + item;
        next.write(sum);
    }
}
```

Student ID: ....

(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]

```
        Buffer b = new Buffer(2);
        b.write(1);
        System.out.println(b.read());
```

(ii) [2 marks]

```
        Buffer b = new Buffer(2);
        b.write(20);
        b.write(30);
        System.out.println(b.read());
```

(iii) [2 marks]

```
        Buffer b = new Buffer(2);
        Pipe p = new NegativeFilter(b);
        p.write(-99);
        p.write(100);
        System.out.println(b.read());
```

(iv) [2 marks]

```
    public static void question1d() {
        Buffer b = new Buffer(10);
        Pipe nf = new NegativeFilter(b);
        Pipe acc = new Accumulator(nf);
        acc.write(100);
        acc.write(-99);
        acc.write(10);
        System.out.println(b.read() + "," + b.read());
```

(**b**) Consider the following error message:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
        at Buffer.write(Buffer.java:9)
```

(**i**) [3 marks] Briefly, describe how this error could have arisen.

(**ii**) [3 marks] In the box below, provide code which will cause this error.

(**c**) [6 marks] `Fork` is an implementation of `Pipe` which connects to *two* `Pipe` instances. When an item is written to a `Fork`, it is then written to both connections. In the box below, provide an implementation of `Fork`.

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.

(d) [5 marks] In the box below, explain the meaning of the following statement in terms of *objects* and *references*. Your discussion should include a diagram to illustrate.

"A pipeline consists of one or more pipes connected together, and ending with a buffer"

(e) [5 marks] In the box below, explain the meaning of the following statement. Your discussion should indicate how you would modify the code given on page 2.

"Abstract classes can be used to eliminate duplicate code"

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. Java Generics [30 marks]

(a) Recall the `Pipe` interface from page 2, and its associated implementations.

(i) [2 marks] By writing neatly in the box below, turn `Pipe` into a generic version `Pipe<T>` where `T` specifies the type of items which can be written.

```
interface Pipe {
    void write(int item);
}
```

(ii) [6 marks] By writing neatly in the box below, turn `Buffer` into a generic version `Buffer<T>`.

```
class Buffer implements Pipe {

    private int[] items;

    private int writePos;

    private int readPos;

    public Buffer(int len) {

        items = new int[len];
    }

    public void write(int x) {

        items[writePos] = x;

        writePos = writePos + 1;
    }

    public int read() {

        int x = items[readPos];

        readPos = readPos + 1;

        return x;
    }
}
```

(b) The following interface illustrates a "colour pipeline". That is, a kind of `Pipe` to which we can only write instances of the class `Colour`.

```
interface ColourPipeline<T extends Colour> extends Pipe<T> { }
```

(i) [4 marks] In the above, "`<T extends Colour>`" indicates that `Colour` is an *upper bound* on `T`. In your own words, briefly explain what this means.

(ii) [5 marks] Provide a *generic method* `writeAll(T[],ColourPipe<T>)` in the box below which writes every array element into the `ColourPipe<T>` parameter.

Student ID: ....

(c) The following code does not compile because `Pipe<Colour>` is not a *subtype* of `Pipe<Object>`.

```
1 Pipe<Colour> pipeCol = new Buffer<Colour>(10);
2 Pipe<Object> pipeObj = pipeCol;
3 pipeObj.write("Hello");
```

(i) [2 marks] Identify the line number above to which the compilation error would refer.

(ii) [3 marks] Suppose the Java compiler allowed the above program to compile. What problem would arise when executing the program?

(d) This question concerns Java's *wildcard types* (e.g. `Pipe<?>`).

(i) [3 marks] Briefly, explain what the wildcard `?` in type `Pipe<?>` means.

(ii) [5 marks] Briefly, explain why the type `Pipe<String>` is a subtype of `Pipe<?>`.

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. 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]

```java
//The answer must have balanced parenthesis
class Hi{
    public String speak(){return "hi";}
}
class Hello{
    public String speak(){return "hello";}
}
class LoudHi extends Hi{
    [???]
}
class LoudHello extends Hello{
    [???]
}

public class Q1 {
  public static void main(String[]arg){
    assert new LoudHello().speak().equals("hello!!");
    assert new LoudHi().speak().equals("hi!!");
  }
}
```

(b) [5 marks]

```
//The answer must have balanced parenthesis
import java.util.*;

class Printer{
    public String sep(){return "@";}
    final public <T> String format(List<T> l){
        String res="";
        for(T t : l){res += t + sep();}
        return res;
    }
}
[???]
public class Q2 {
  public static void main(String[]arg){
  Printer p=new HashPrinter();
  assert p.format(Arrays.asList(1,2,3)).equals("1#2#3#");
  }
}
```

(c) [5 marks]

```java
//The answer must have balanced parenthesis
class A{
    public int f;
    A(int f){this.f=f;}
}
[???]

public class Q3 {
  public static void main(String[]arg){
   B b=new B(2,"Hi");
   A a=b;
   assert a.f==2 && b.g.equals("Hi");
  }
}
```

(d) [5 marks]

```
//The answer must have balanced parenthesis

interface Counter{
  int nextNum();
}

public class Q4 {
  public static void main(String[] arg){
      Counter c1=[???];
      assert "10,11".equals(c1.nextNum()+","+c1.nextNum());
      assert "12,13".equals(c1.nextNum()+","+c1.nextNum());
  }
}
```

(d) [6 marks] Using assertions, rewrite the code below by adding runtime checks which verify the pre/post conditions:

- Precondition: `l!=null`
- Postcondition:

  if `res==indexOf(elem,l)`, then `l.get(res).equals(elem)`

  if `indexOf(elem,l)` throws `ElementNotFound`, then no `i` exists such that `l.get(i).equals(elem)`

```
int indexOf(String elem,List<String>l){
  for(int i=0;i<l.size();i++){
    if(elem.equals(l.get(i))){return i;}
  }
  throw new ElementNotFound(elem+"␣not␣present␣in"+l);
}
```

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

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.