THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

This exam paper must not be removed from the venue

Venue	____________________
Seat Number	________
Student Number	|__|__|__|__|__|__|__|__|
Family Name	_____________________
First Name	_____________________

School of Information Technology and Electrical Engineering

EXAMINATION

Semester Two Final Examinations, 2020

CSSE2002 Programming in the Large

This paper is for St Lucia Campus students.

Examination Duration: 120 minutes

Reading Time: 10 minutes

Exam Conditions:

.Set start and completion time for all students e.g. a 2 hour exam, starts at 8am, ends at 10am

Paper-based exam (on-campus exam only)

This is a Closed Book examination - specified written materials permitted

No calculators permitted

Materials Permitted In The Exam Venue:

(No electronic aids are permitted e.g. laptops, phones)

Blank scrap paper permitted - two A4 sheets permitted

One A4 sheet of handwritten or typed notes double sided is permitted

Materials To Be Supplied To Students:

1 x 14-Page Answer Booklet

1 x Multiple Choice Answer Sheet

Instructions To Students:

Additional exam materials (eg. answer booklets, rough paper) will be provided upon request.

Answer all questions. Marks are indicated for each question.

For Examiner Use Only

Question	Mark
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

Total ________

Question 1. 4 marks

Which of the following are valid initialisations? (select all that apply)

a) List<String> l = new ArrayList<String>();

b) List<Double> l = new List<Double>();

c) List<int> l = new ArrayList<int>();

d) LinkedList<Exception> l = new LinkedList<>();

e) ArrayList<String> l = new LinkedList<String>();

Question 2 4 marks

What will the output be when the main() method is executed?

```
public class SwitchStatements {
    private static int count;
    private static String result = "";
    public static String getNumber(int number) {
        count += 1;
        switch (count) {
            case 0:
                result += "Zero";
                break;
            case 1:
                result += "One";
                break;
            case 2:
                result += "Two";
                break;
            case 3:
                result += "Three";
                break;
            default:
                result += "Unknown";
                break;
        }
        return result;
    }

    public static void main(String args[]) {
        count = 1;
        getNumber(count);
        count -= 3;
        getNumber(count);
        count += 4;
        System.out.println(getNumber(count));
    }
}
```

a) The code would throw an exception

b) TwoZeroUnknown

c) OneUnknownTwo

d) TwoUnknownUnknown

e) TwoOneUnknown

Question 3 4 marks

What will the output be when the main() method is executed?

```
public static void main(String[] args) {
    TreeMap<Integer, String> values = new TreeMap<Integer, String>();
    values.put(40, "Fun");
    values.put(10, "Hello");
    values.put(30, "Java");
    values.put(60, "Program");
    values.put(20, "World");
    values.put(50, "Code");

    Set set = values.entrySet();
    Iterator it = set.iterator();
    while(it.hasNext()) {
        System.out.println(((Map.Entry) it.next()).getKey() + "," +
                ((Map.Entry) it.next()).getValue());
    }
}
```

a) 40,Fun
30,Java
20,World

b) The program would throw a runtime exception

c) 10,Hello
20,World
30,Java
40,Fun
50,Code

d) 10,World
30,Fun
50,Program

e) 40,Fun
10,Hello
30,Java
60,Program
20,World
50,Code

Question 4 4 marks

What will the output be when the main() method is executed?

```
abstract class R {
    int num = 1;
    public int doSomething(int a) {
        this.num += 1;
        return this.num * a;
    }
}

class S extends R {
    @Override
    public int doSomething(int a) {
        this.num += 1;
        return this.num * super.doSomething(a);
    }
}

class T extends S {
    int num = 1;
    @Override
    public int doSomething(int a) {
        this.num += 1;
        return this.num * super.doSomething(a);
    }
}

public class Inheritance {
    public static void main(String... args) {
        ArrayList<R> l = new ArrayList<>();
        l.add(new S());
        l.add(new T());
        for (R item : l) {
            System.out.println(item.doSomething(1));
        }
    }
}
```

a) 6
   12

b) 4
   8

c) 6
   4

d) 6
   24

e) The code would not compile.

Question 5 5 marks

What will the output be when the main() method is executed?

```
public static void main(String[] args) {
    List<Integer> numbers = new ArrayList<>();

    List<Integer> newNumbers = new ArrayList<>();
    for (int i = 0; i <= 12; i++) {
        numbers.add(i * 2);
    }

    for (int i = 0; i < numbers.size(); i++) {
        if (numbers.get(i) % 3 == 0) {
            newNumbers.add(numbers.get(i));
        }
        if (numbers.get(i) % 4 == 0) {
            newNumbers.add(numbers.get(i));
        }
    }
    System.out.println(newNumbers);
}
```

a) [0, 4, 6, 8, 12, 16, 18, 20, 24]

b) [0, 0, 4, 6, 8, 12, 12, 16, 18, 20]

c) [4, 6, 8, 12, 12, 16, 18, 20, 24, 24]

d) [0, 0, 4, 6, 8, 12, 12, 16, 18, 20, 24, 24]

e) The code would generate a run time exception

Question 6 5 marks

What will the output be when the main() method is executed?

```
public static void main(String args[]) {
    int[] a = {1, 3, 5, 7, 9};
    int[] b = {2, 4, 6, 8, 10};
    method(b, a);
    System.out.println(b[0] + "," + b[1] + "," + b[2] +
            "," + b[3] + "," + b[4]);
    System.out.println(a[0] + "," + a[1] + "," + a[2] +
            "," + a[3] + "," + a[4]);
}

public static void method(int[] a, int [] b) {
    for (int i = 0; i < a.length; i++) {
        if (i % 2 == 1) {
            b[i] = a[i];
        } else {
            a[i] = b[i] * -1;
        }
    }
}
```

a) 2,3,6,7,10
   2,-3,6,-7,10

b) 1,4,5,8,9
   -1,4,-5,8,-9

c) 1,-4,5,-8,9
   1,4,5,8,9

d) 2,3,6,7,10
   -2,3,-6,7,-10

e) -1,4,-5,8,-9
   1,4,5,8,9

Question 7 10 marks

Implement the loadNamesFromFile() method according to the Javadoc.

```
/** Open the file with the given filename, extract the names from each
  * line, then return a list containing all the names in the file.
  * Each line in the textfile uses the following format:
  * idnumber,name,occupation
  * e.g.
  * 1,John,Student
  * 2,Sophie,Researcher
  *
  * You can assume that the input file will always be in the correct
  * format.
  * You should assume that the text file can hold details for ANY number
  * of people, as long as it is at least one person.
  * The method must use the FileReader class.
  * The method must catch all exceptions. Catching the generic
  * Exceptions class is not allowed.
  * The method cannot declare that it throws any exceptions.
  *
  * @param filename the name of the file to be opened. Must be
  *        in .txt format.
  * @return list of names in the file
  */
public static List<String> loadNamesFromFile(String filename) {

    // implement this function

}
```

Question 8 12 marks

Assume that the following four classes are all in separate files within the same package.
Do these classes exhibit high or low coupling?
Provide a detailed justification for your answer.
Your answer should cover all aspects of the coupling in the code.

```
public class X {
    public int num = 5;
    protected Z z;

    public X(Z z) {
        this.z = z;
    }

    public void doThis() {
        sayHello();
    }

    public void sayHello() {
        System.out.println("Hello");
    }
}

public class Y extends X {
    public Y(Z z) {
        super(z);
    }

    public void doThat() {
        this.z.sayHello();
    }

    @Override
    public void sayHello() {
        super.sayHello();
    }

}

public class Z {
    private X x = new X();

    public void setNum(int num) {
        x.num = num;
    }

    public void sayHello() {
        System.out.println("Hello");
    }
}
```

Question 9 7 marks

Provide a complete set of black box test cases for the `timesLetterInString` method.

For each test case, provide the input, expected output and a justification for each case.

Do **not** write JUnit tests.

```
/**
 * @require toSearch != null && searchingFor != ''
 *
 * @ensure \result number of times the letter appears in the string.
 *         the result is case sensitive
 */
public static int timesLetterInString(String toSearch,
            char searchingFor) {
    int total = 0;
    for (int i = 0; i < toSearch.length(); i++) {
        if (toSearch.charAt(i) == searchingFor) {
            total++;
        }
    }
    return total;
}
```

Question 10 7 marks

Provide a complete set of white box test cases for the `timesLetterInString` method in question 9.

For each test case, provide the input, expected output and a justification for each case.

You may have some overlap in the inputs and expected outputs for the white box test cases and the black box test cases from question 9.

Do **not** write JUnit tests.

Question 11 14 marks

Write code to implement a generic Port class that can only hold a Ship or a specified subtype of Ship.

The class must have:

- a constructor
- a method to add a Ship to the Port
- a method to remove a Ship from the Port
- a method to retrieve all Ships which are of the same type as a provided Ship

The method to retrieve all Ships which are of a given type must be provided with a parameter of a *generic type*. You must use the getClass() method to evaluate if each ship in the Port is of the given type.

You must use collections of the **List** type (or subtypes of List).

Question 12 11 marks

Write a **recursive** method `reverseWordsInString` that given a string as a parameter, will reverse the order of the words in the string, and return the result.

E.g. Passing “this is a sentence” as the parameter would return “sentence a is this”.

You cannot not use the split() method or regular expressions, in any way.

All other methods are allowed though.

Question 13 5 marks

Write a Javadoc comment for the `reverseWordsInString` method which was implemented in Question 12. The Javadoc should describe the method and provide all the tags that are appropriate for the method's signature.

Question 14 8 marks

The Liskov substitution principle requires that a subtype's overridden methods must satisfy the supertype's method specifications. It guarantees that code expecting the supertype method behaves correctly even though the calls go to the subtype implementation.

```
class E {
    /**
     * @require areas != null && sorted(areas) && minArea < maxArea
     *          && minvalue(area) > 3
     * @ensure  \forall int i; 0 <= i && i < areas.size();
     *              \result is the sum of all the areas where
     *              minArea <= areas[i] && areas[i] < maxArea
     */
    public double sumAllAreasWithinRange(List<Integer> areas,
            int minArea, int maxArea) { }
}

class F extends E {
    /**
     * @require areas != null && sorted(areas) && minArea < maxArea
     *          && minvalue(area) > minArea
     * @ensure  \forall int i; 0 <= i && i < areas.size();
     *              \result is the sum of all the areas where
     *              minArea <= areas[i] && areas[i] <= maxArea
     */
    public double sumAllAreasWithinRange(List<Integer> areas,
            int minArea, int maxArea) { }
}

class G extends E {
    /**
     * @require areas != null && sorted(areas) && minArea <= maxArea
     *          && minvalue(area) > 3
     * @ensure  \forall int i; 0 <= i && i < areas.size();
     *              \result is the sum of all the areas where
     *              minArea < areas[i] && areas[i] < maxArea
     */
    public double sumAllAreasWithinRange(List<Integer> areas,
            int minArea, int maxArea) { }
}
```

Does class **F** satisfy the substitution principle with respect to class **E**? Explain why or why not.

Does class **G** satisfy the substitution principle with respect to class **E**? Explain why or why not.

END OF EXAMINATION