

Xi'an Jiaotong-Liverpool University

西交利物浦大學

PAPER CODE	EXAMINER	DEPARTMENT	TEL
CSE210		Computer Science & Software Engineering	

SECOND SEMESTER 2018/2019 FINAL EXAMINATIONS

BACHELOR DEGREE – Year 3

Advanced Object Oriented programming

TIME ALLOWED: 2 Hours

---

**INSTRUCTIONS TO CANDIDATES**

- 1、 Total marks available are 100. This will count for 50% in the final assessment.
- 2、 Answer all FOUR questions.
- 3、 The number in the column on the right indicates the marks for each section.
- 4、 Answer should be written in the answer booklet(s) provided.
- 5、 The university approved calculator - Casio FS82ES/83ES can be used.
- 6、 All the answers must be in English.

**Question 1.** Consider the following three classes in the same package: Point, LinkedList and Node. Node is the inner class of LinkedList. Answer the following questions.

[25 marks]

```
public class Point{
    final int xCoord;
    final int yCoord;

    public Point(int xCoord, int yCoord){
        this.xCoord = xCoord;
        this.yCoord = yCoord;
    }

    public int getXCoord{
        return xCoord;
    }

    public int getYCoord(){
        return yCoord;
    }

    protected int getMax(){
        return Math.max(xCoord,yCoord);
    }
}

public class LinkedList{
    private Node head;

    public LinkedList(){
        head = null;
    }

    public void addToEnd(Point point){
        // implement your code here
    }

    private class Node{
        private Point point;
        private Node next;

        public Point getPoint(){
            return point;
        }
    }
}
```

```
        public Node getNext(){
            return next;
        }
    }
}
```

- a) The two fields, `xCoord` and `yCoord` are defined as `final` in the `Point` class. Are they visible to the `LinkedList` class and `Node` class?

[2 marks]

- b) The method `getMax()` is defined as `protected` in the `Point` class. Is it visible to the `LinkedList` class and `Node` class?

[2 marks]

- c) The field `head` is defined as `private` in the `LinkedList` class, is it visible to the `Point` class and the `Node` class?

[2 marks]

- d) Provide a constructor for the class `Node`, which can be used to create a `Node` instance with a point and a node.

[4 marks]

- e) Reuse the `Node` constructor from question 1.d), and implement the method `addToEnd()` for the `LinkedList` class which inserts a given node to the end of the list.

[6 marks]

- f) The Java interface `Comparable` provides a method `compareTo(Object o)` that allows comparison between objects.

```
public interface Comparable{
    public int compareTo(Object o);
}
```

The method returns "0" if two instances are equal. It returns a positive number if the current instance is greater than the "Object o" and a negative number if less than the "Object o". For the `Point` class, assume that the criteria for the comparison is the Euclidean distance to the origin. Implement the `compareTo()` method for the class `Point` to make its instances comparable.

[5 marks]

- g) Describe how to modify the class `Point` so that it can keep track of the largest `xCoord` ever created.

[4 marks]

Question 2. Answer the following questions.

[25 marks]

- a) Consider the class BSTree below which implements a binary search tree. Describe the state of the method-call stack when `System.out.println(bt.isInOrder())` is executed in the `main()` method.

[6 marks]

```
public class BSTree {

    private BSTree leftSubtree;
    private int value;
    private BSTree rightSubtree;

    BSTree(BSTree left, int val, BSTree right) {
        leftSubtree = left;
        value = val;
        rightSubtree = right;
    }

    public int getValue() {
        return value;
    }

    public boolean isInOrder() {
        if (leftSubtree == null) {
            return value < rightSubtree.getValue();
        } else if (rightSubtree == null) {
            return value > leftSubtree.getValue();
        } else {
            return leftSubtree.getValue() < value && value <
rightSubtree.getValue();
        }
    }

    public static void main(String[] args) {
        BSTree bt = new BSTree(null, 5, null);
        bt = new BSTree(bt, 7, null);
        bt = new BSTree(null, 6, bt);
        System.out.println(bt.isInOrder());
    }
}
```

- b) In the context of Java programming, how can one check if an asserted property is a class invariant?

[2 marks]

- c) Assume that a class invariant for Binary Search Tree is defined as: (1) each value in the left subtree is strictly less than the internal value; and (2) each value in the right subtree is strictly greater than the internal value. Does the implementation above preserve the class invariant? Justify your answer with an example.

**[6 marks]**

- d) What is the main difference between a checked exception and an unchecked exception?

**[4 marks]**

- e) Class `Client` implements a client program which connects and sends messages to a server class called `EchoServer`. The server receives the message and simply echoes it back to the client. In the `EchoServer` class, an `IOException` may be thrown in two circumstances: 1) the `accept()` method of the `ServerSocket` class; and 2) reading input from the client. Discuss how these two cases would be handled in order for the `EchoServer` implementation to be robust.

**[4 marks]**

- f) Describe how a multi-threaded server keeps track of the number of active client connections.

**[3 marks]**

**Question 3.** Consider the bounded queue class, BoundedQueue, which can store at most 10 integers. Answer the following questions.

[25 marks]

```
public class BoundedQueue {
    private int[] items = new int[10];
    private int first = -1;
    private int last = 0;

    public boolean isEmpty() {
        return last <= 0;
    }

    public int get(){
        int val = items[first];
        for (int i = 0; i + 1 < last; i++) {
            items[i] = items[i + 1];
        }
        last--;
        if (last == 0) {
            first--;
        }
        return val;
    }

    public void add(int i) {
        if (last == 0) {
            first++;
        }
        items[last++] = i;
    }

    public void remove(int i) {
        if (i <= 0) {
            return;
        }
        get();
        remove(i - 1);
    }

    public static void main(String[] args) {
        BoundedQueue s = new BoundedQueue();
        s.add(3);
        s.remove(3);
    }
}
```

- a) Briefly describe what will happen when the `main()` method of the `BoundedQueue` is executed.  
[2 marks]
- b) Write a checked exception called `QueueEmptyException` for the class `BoundedQueue` by overriding the `getMessage()` method.  
[4 marks]
- c) Modify the `get()` method so that it can throw an `QueueEmptyException` when the stack is empty.  
[6 marks]
- d) With the modification of the `get()` method, what other changes would be needed for the class `BoundedQueue`?  
[4 marks]
- e) Class `ThreadBoundedQueue` creates 3 threads to access a shared resource, which is an instance of class `BoundedQueue`. The `main()` method is supposed to print integers from 0 to 29, once for each integer. Will the problem of interference occur during the execution? Justify your answer with an example.  
[5 marks]

```
class ThreadBoundedQueue implements Runnable {
    private BoundedQueue q;
    private static int count = 0;

    ThreadBoundedQueue(BoundedQueue q) {
        this.q = q;
    }

    public void run() {
        for (int i = 0; i < 20; i++) {
            if (q.isEmpty()) {
                q.add(count++);
            } else {
                try {
                    System.out.println(q.get());
                } catch (Exception e) {
                    System.err.print(e.getMessage());
                }
            }
        }
    }
}
```

```
public static void main(String[] args) {  
    BoundedQueue q = new BoundedQueue();  
    Thread t1 = new Thread(new ThreadBoundedQueue(q));  
    Thread t2 = new Thread(new ThreadBoundedQueue(q));  
    Thread t3 = new Thread(new ThreadBoundedQueue(q));  
    t1.start();  
    t2.start();  
    t3.start();  
}  
}
```

- f) With your answer from Question 3.e), what changes would be necessary for the ThreadBoundedQueue class?

[4 marks]



Question 4. Answer the following questions.

[25 Marks]

- a) Define an interface called `Operation` using Java Generics. It should have two methods, `perform()` and `name()`. The `perform()` method takes an input of the type `A` and returns an output of the type `B`. The `name()` method takes no parameter and returns a `String`.

[4 marks]

- b) Briefly describe what an abstract class is and its characteristics.

[4 marks]

- c) Define an abstract class called `AbstractOperation` using Java Generics which implements the `Operation` interface. You should provide a constructor which can assign a name to an operation, and implement the `name()` method. The `perform()` method is abstract.

[5 marks]

- d) Define a subclass called `ReplicateFunction` of the class `AbstractOperation`. The class should implement the `perform()` method, which takes a string as input and returns the duplicated string of the input.

[5 marks]

- e) What is an anonymous class and what is its advantage?

[2 marks]

- f) Consider the following class `Replicate2XFunction`. What is the output after the `main()` method is executed?

[5 marks]

```
1. public class Replicate2XFunction {
2.     public Operation compose(final ReplicateFunction rf){
3.         return new AbstractOperation<String, String>(){
4.             public String perform(String s){
5.                 return rf.perform(rf.perform(s));
6.             }
7.         }
8.     public static void main(String args[]){
9.         Replicate2XFunction r2f = new Replicate2XFunction();
10.        Operation<String, String> op = r2f.compose(new
ReplicateFunction("REFUN"));
11.        System.out.println(op.perform("CSE210"));
12.    }
13.}
```

**END OF EXAM PAPER**