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 RESIT 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 two different packages, `Person`, `LinkedList` and `Node`. Class `Node` is an inner class of `LinkedList`. Answer the following questions.

**[25 marks]**

```
public class Person{
    private final String name;
    private int age;
    static int maxAge;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    public void updateName(String name){
        this.name = name;
    }
}

public class LinkedList{
   private Node start;
   private Node end;

   public LinkedList(){
      start = null;
      end = null;
   }

   public void addToStart(Person p) {
    start = new Node(p, start);
    if (listEnd == null) {
      end = start;
    }
  }

   public void addToEnd(Person p){
      // implement your code here
   }

    private class Node{
       private Person p;
       private Node next;

        Node(Person person, Node next){
            this.person = person;
```

```
            this.next = next;
        }

        public Person getPerson(){
            return p;
        }

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

a) The `maxAge` field defined in the `Person` class has a default scope. Is it visible to the `LinkedList` and `Node` classes, respectively?

**[2 marks]**

b) Are the private fields, `start` and `end`, defined in the `LinkedList` class, visible to the `Person` and `Node` classes, respectively?

**[2 marks]**

c) Will the `updateName()` method in the `Person` class be able to update the name of an existing `Person` instance? Justify your answer.

**[4 marks]**

d) Modify the constructor of the `Person` class, so that `maxAge` can store the maximum age for all `Person` instances ever created.

**[5 marks]**

e) Implement the `addToEnd()` method for the `LinkedList` class. Existing constructor and methods can be used where appropriate.

**[6 marks]**

f) A new method called `removeFromEnd()` can be defined for the `LinkedList` class, which removes the last node in the list. Write the pseudocode for the method.

**[6 marks]**

**Question 2.** The class `ShopQueue` implements a bounded queue for a small shop. The queue can accommodate at most 5 guests at any given time. The staff of the shop calls `addGuest()` method if a guest can be added into the queue. If a guest is serviced, the staff calls the `getNextGuest()` or `remove()` method. Assume that the method calls can only be performed one by one. Answer the following questions.

**[25 marks]**

```
public class ShopQueue {
    private int[] guests = new int[5];
    private int top = 0;

    public int getNextGuest(){
        int first = guests[0];
        for (int i = 0; i + 1 < top; i++) {
            guests[i] = guests[i + 1];
        }
        top--;
        return first;
    }

    public void addGuest(int i) {
        guests[top++] = i;
    }

    public void remove() {
        System.out.println(getNextGuest());
    }

    public static void main(String[] args) {
        ShopQueue sq = new ShopQueue ();
        sq.addGuest(1);
        sq.addGuest(3);
        sq.remove();
    }
}
```

a) Describe the state of the method-call stack during execution of the `main()` method.

**[6 marks]**

b) What is meant by class invariant in the context of Java programming?

**[2 marks]**

c) Is 'top >= 0' a class invariant for the `ShopQueue` class? Justify your answer.

**[3 marks]**

d) Define a checked exception class `EmptyQueueException` for the `ShopQueue` by overriding the `getMessage()` method.

**[4 marks]**

e) Modify the `getNextGuest()` method so that it can throw an `EmptyQueueException` when the queue is empty.

**[6 marks]**

f) With the modified `getNextGuest()` method, what other changes would be necessary for the `ShopQueue` class?

**[4 marks]**

**Question 3.** Consider the `MessagePrinter` class below and answer the following questions.

**[25 marks]**

```
class MessagePrinter implements Runnable {
    private String message;

    MessagePrinter(String s) {
        message = s;
    }

    public void run() {
        for (int i = 0; i < 100; i++) {
            for (int j = 0; j < message.length(); j++) {
                System.out.print(message.charAt(j));
            }
            System.out.print("\n");
        }
    }

    public static void main(String[] args) {
        MessagePrinter mp1 = new MessagePrinter("Hello, OOP");
        MessagePrinter mp2 = new MessagePrinter("Goodbye, OOP");
        Thread t1 = new Thread(mp1);
        Thread t2 = new Thread(mp2);
        t1.start();
        t2.start();
    }
}
```

a) In the context of Java multithreading, what is meant by interference?

**[4 marks]**

b) Discuss in detail how the interference problem may occur in class `MessagePrinter`.

**[5 marks]**

c) What change would be needed to the `MessagePrinter` class to solve the interference problem?

**[6 marks]**

d) Briefly describe how a running thread may move to other states, i.e., Ready, Sleeping, Waiting, Dead and Blocked.

**[5 marks]**

e) In the context of multithreading, what is meant by deadlock?

**[3 marks]**

f) How to prevent deadlock in Java multi-threading programming?

**[2 marks]**

**Question 4.** Answer the following questions.

**[25 Marks]**

a) Briefly explain the term 'Generics' in Java and its benefits.

**[2 marks]**

b) Define a generic interface called `PrimitiveFunction`. The interface needs to have a method `compute()`, which accepts a parameter of type X, and returns a type Y.

**[4 marks]**

c) Define an interface called `CompositeFunction` as a sub-interface of PrimitiveFunction. The interface needs to define an extra method called `name()`, which accepts no parameter, and returns a `String` type.

**[4 marks]**

d) Implement a class called `LengthFunction` by implementing the `PrimitiveFunction` interface. The `compute()` method computes the length of a given string.

**[4 marks]**

e) Define a class called `Mod5Function` by implementing the `PrimitiveFunction` interface. The `compute()` method for this class takes an `Integer` as input and return a `Boolean` value indicating if result of "modulus 5" is zero.

**[4 marks]**

f) What is an anonymous class and what is its main advantage?

**[2 marks]**

g) Consider the class `ComposeFunction` below, which makes use of an anonymous class. What is the output after the `main()` method is executed?

**[5 marks]**

```
public class ComposeFunction {

    public CompositeFunction compose(final PrimitiveFunction<String,
Integer> f1, final PrimitiveFunction<Integer, boolean> f2, final String
name){
        return new CompositeFunction<String, boolean>(){
            public boolean compute(String s){
                return f2.compute(f1.compute(s));
            }

            public String name(){
                return name;
            }};
    }
```

```
    public static void main(String args[]){
        ComposeFunction f = new ComposeFunction();
        CompositeFunction<String,  boolean>  cf  =  f.compose(new
LengthFunction(), new Mod5Function (), "Length&Mod");
        System.out.println(cf.name());
        System.out.println(cf.compute("ILoveProgramming"));
    }
}
```

**END OF EXAM PAPER**