NEWCASTLE UNIVERSITY

SEMESTER 2 2014/2015

ADVANCED PROGRAMMING

Time allowed – 3 Hours

Instructions to candidates:

Answer TWO questions from Section A

and TWO questions from Section B.

The total marks available for this exam are 100

Marks shown for sub-sections are indicative only

[Turn Over]

SECTION A

Answer TWO questions in this Section.

Question A1

a) Describe defensive copying in terms of a reference parameter to a constructor of a class. [2 marks]

b) What is an immutable class? Give three advantages of immutable classes. [4 marks]

c) Consider the following Tutor and Student classes:

```
public class Student {
    private String name;
    private String course;

    public Student(String name, String course){
        this.name = name;
        this.course = course;
    }

    public String getName() { return name; }

    public String getCourse() { return course; }

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

    public void setCourse(String course){
        this.course = course;
    }
}
```

```
public final class Tutor {
    private String name;
    private final Set<Student> tutees;

    public Tutor(String name, Student[] students){
        this.name = name;
        tutees = new HashSet<Student>();
        for (int i = 0; i<students.length; i++)
            tutees.add(students[i]);
    }

    public Set<Student> getTutees(){
        return Collections.unmodifiableSet(tutees);
    }

    public String getName() { return name; }

}
```

i) Is it possible to extend (i.e. write a subclass of) the `Tutor` class? Why? [2 marks]

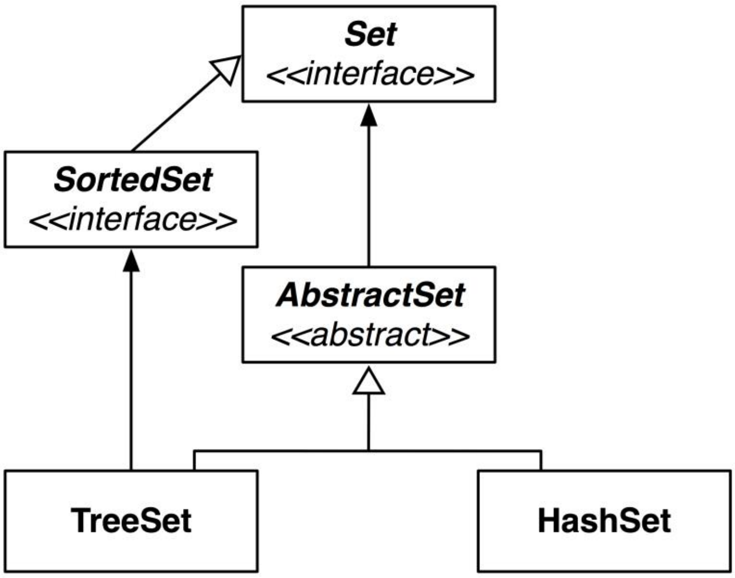
ii) Explain why instances of both `Student` and `Tutor` are mutable. [3 marks]

iii) Re-write the `Tutor` class to make it immutable (without modifying the `Student` class). [10 marks]

iv) Briefly describe the changes necessary to make `Student` immutable. If these changes were made, would it still be necessary to re-write the `Tutor` class as you did in iii) above? Briefly explain your answer. [4 marks]

Question A2

Consider the following diagram that shows the `Set` inheritance hierarchy of the Java collections framework.



a) Briefly explain the relationships between interface, abstract and implementation classes in the hierarchy. [4 marks]

b) You have been asked to record the number of attempts to remove elements from a `TreeSet`. Given the following extract from the `Set` interface:

```
public interface Set<E> {

    boolean add(E element);
    boolean addAll(Collection<? extends E> c);
    boolean contains(Object o);
    boolean containsAll(Collection<?> c);
    boolean isEmpty();
    boolean remove(Object o);
    boolean removeAll(Collection<?> c);
    int size();
    …
}
```

i) Extend `TreeSet` to provide an `InstrumentedTreeSet` that records the number of attempts to remove elements from the set. Your `InstrumentedTreeSet` must provide the following method that is not provided by `TreeSet`:

- `getRemoveCount` that returns the number of attempts to remove elements from the set.

Your implementation of `InstrumentedTreeSet` should show:

- the class declaration,
- the declaration of a member field to record attempts to remove elements,
- implementations of the methods in the extract of the `Set` interface that must be overridden, and
- an implementation of `getRemoveCount.`

You do not need to define `InstrumentedTreeSet` constructors. [10 marks]

ii) Why does the use of inheritance to instrument the `TreeSet` become a problem when we consider instrumenting other `Set` implementations in the same way? Explain how composition can address this problem. Give another advantage of the use of composition (as opposed to implementation inheritance). [5 marks]

iii) Provide a partial implementation of an `InstrumentedSet` that uses interface inheritance and composition to instrument any implementation of `Set`. Your partial implementation must show:

- the `InstrumentedSet` class declaration,
- member field declarations,
- the implementation of the `remove` method, and the implementation of some other method of the `Set` interface that does not require any instrumentation, e.g. `add`. [6 marks]

Question A3

a) A hash table data structure can be used to implement a set or a map.

i) With the aid of a diagram, describe the structure of a hash table that is used to implement a set. Explain how the structure enforces the no duplicate elements invariant of sets. [7 marks]

ii) Under normal conditions, a hash table provides fast access to its elements (e.g. add and remove are O(1) operations). Explain why this is and under what conditions (and how) performance of a hash table degrades. [3 marks]

iii) Explain why the inconsistent implementation of Object equals and hashCode methods can undermine the integrity of Java hash table data structures. [2 marks]

b) Provide an implementation of the compareTo method for the following Name class that orders instances of Name by lastName then firstName.

```
public final class Name implements Comparable<Name>
    private final String firstName;
    private final String lastName;
    public Name(String firstName, String lastName){
        this.firstName = firstName;
        this.lastName = lastName;
    }
    public String getFirstName(){ return firstName;}
    public String getLastName(){ return lastName;}
    public int compareTo(Name name){
        // provide implementation of this method
    }
}
```

[4 marks]

c) Provide a `Comparator` that orders instances of `Name` by `firstName` then `lastName`. [2 marks]

d) Two possible implementations of the List data structure are a linked list and a resizable array. For each of the following scenarios, state which of these two implementations would be the most efficient to use and why.

i) A queue of people at a cinema ticket office.

ii) The names in a mobile phone address book. [4 marks]

e) A Set would be a more suitable data structure than a List for the names in a mobile phone address book, the scenario from d) ii) above. State the reason for this and also which of the Java set implementations (HashSet or TreeSet) you would use if you were to implement this and why. [3 marks]

SECTION B

Answer TWO questions in this Section.

Question B1

a) Outline the steps that a Java programmer should take in order to define and start a new thread of concurrent activity. [5 marks]

b) Write a multithreaded program containing the main thread and two additional threads: one to print the integers from 1 to 100 and the second one to print the letter A 40 times. Design your program in such a way that the numbers from 51 to 100 are printed after all letters were already printed. [15 marks]

Hint: The main thread can create the thread printing numbers, which, in turn, can create (and control) the thread printing the letter A.

c) In what ways is it possible for two threads to access the same object, and what steps should be taken to avoid interference? [5 marks]

Question B2

a) What are the three key features of a monitor and how are they used to support concurrent programming? [6 marks]

b) Explain how the monitor mechanism is implemented in Java. [9 marks]

c) Write a Java monitor to control access to a car park with a certain capacity. It should only permit cars to enter when the car park is not full and, for consistency, does not permit cars to leave when there are no cars in the car park.

[10 marks]

Question B3

a) What are the distinguishing characteristics of an event driven program as opposed to a conventional program? Provide an overview of the basic architecture of an event driven program, including a description of the event processing mechanism. What is the difference between an event handler and an event dispatcher? [8 marks]

b) Describe briefly the basic steps that a Java programmer needs to take in order to set up event handlers and deal with events. [3 marks]

c) Design a simple event-driven program with a graphical user interface that allows the user to enter the radius and height of a cylinder into text boxes, and then calculates and displays the volume of the cylinder (formula: π x radius x radius x height). Describe how you would construct such a program, by identifying the basic Java Swing components you would use, and specifying how events are handled. Provide outline code for the relevant parts of the application.

N.B. The value of π is defined in the `Math` class of the `java.lang` package as `Math.PI`. [14 marks]

END