



King Fahd University of Petroleum & Minerals
College of Computer Science and Engineering
Information and Computer Science Department
First Semester 101 (2010/2011)

ICS 201 - Introduction to Computing II

Final Exam
Saturday January 29th 2011
Time: 120 minutes

Name:

ID#:

Please circle your section number below:

Section	01	02	03	04
Instructor	Sami	Sukairi	Sami	Sukairi
Day and Time	SMW 8 - 8:50	SMW 9 - 9:50	SMW 10 - 10:50	SMW 13:10 - 14:00

Question #	Maximum	Obtained
1	20	
2	20	
3	15	
4	20	
5	25	
Total	100	

Question 1 [20 Points]

1. What keyword is used to specify that a data member is a class data member (shared among all instances of that class)?
 - (a) final
 - (b) shared
 - (c) public
 - (d) static
2. Which of the following is NOT a key component of object oriented programming?
 - (a) Inheritance
 - (b) Encapsulation
 - (c) Polymorphism
 - (d) Parallelism
3. The Java compiler translates source code into
 - (a) machine code.
 - (b) Assembly code.
 - (c) Byte code.
 - (d) JVM code.
4. Which of the following statements is NOT correct?
 - (a) We can use a `new` operator on `String` to create a "String" object.
 - (b) We can use the `new` operator on `int` to create an "int" object.
 - (c) Variables of type "int" can be assigned a value just after being declared.
 - (d) Variables of type "String" can be assigned a value just after being declared.
5. Which of the following variables contains null?

```
String a;  
String b = new String();  
String c = "";  
String d = "null";
```

 - (a) variable a
 - (b) variable b
 - (c) variable c
 - (d) variable d
6. Which of the following characteristics of an object-oriented programming language restricts behavior so that an object can only perform actions that are defined for its class?
 - (a) Dynamic Binding
 - (b) Polymorphism
 - (c) Inheritance
 - (d) Encapsulation
7. What are valid arguments to the `instanceof` operator?

- (a) a class object and a class type
- (b) any primitive type
- (c) boolean type only
- (d) class types only

8. Which of the following is TRUE?

- (a) In java, an instance field declared **public** generates a compilation error.
- (b) **int** is the name of a class available in the package java.lang
- (c) Instance variable names may only contain letters and digits.
- (d) A class has always a constructor (possibly automatically supplied by the java compiler).
- (e) The more comments in a program, the faster the program runs.

9. You read the following statement in a Java program that compiles and executes.

submarine.dive(depth) ;

What can you say for sure?

- (a) **dive** must be a method.
- (b) **dive** must be the name of an instance field.
- (c) **submarine** must be the name of a class
- (d) **submarine** must be a method.

10. What is a constructor's return type?

- (a) void
- (b) Object
- (c) The class name
- (d) A constructor does not have a return type

Question 2 [20 points]

Write a class called *Sentence* with:

1. One instance variable named *text* of type String,
2. One constructor that takes a String as parameter,
3. One accessor method *getText()*,
4. and three methods:

- a- A recursive method void reverse() that reverses the sentence. For example:

```
Sentence greeting = new Sentence("Hello!");  
  
Greeting.reverse();  
  
System.out.println(greeting.getText()); // prints "!olleH".
```

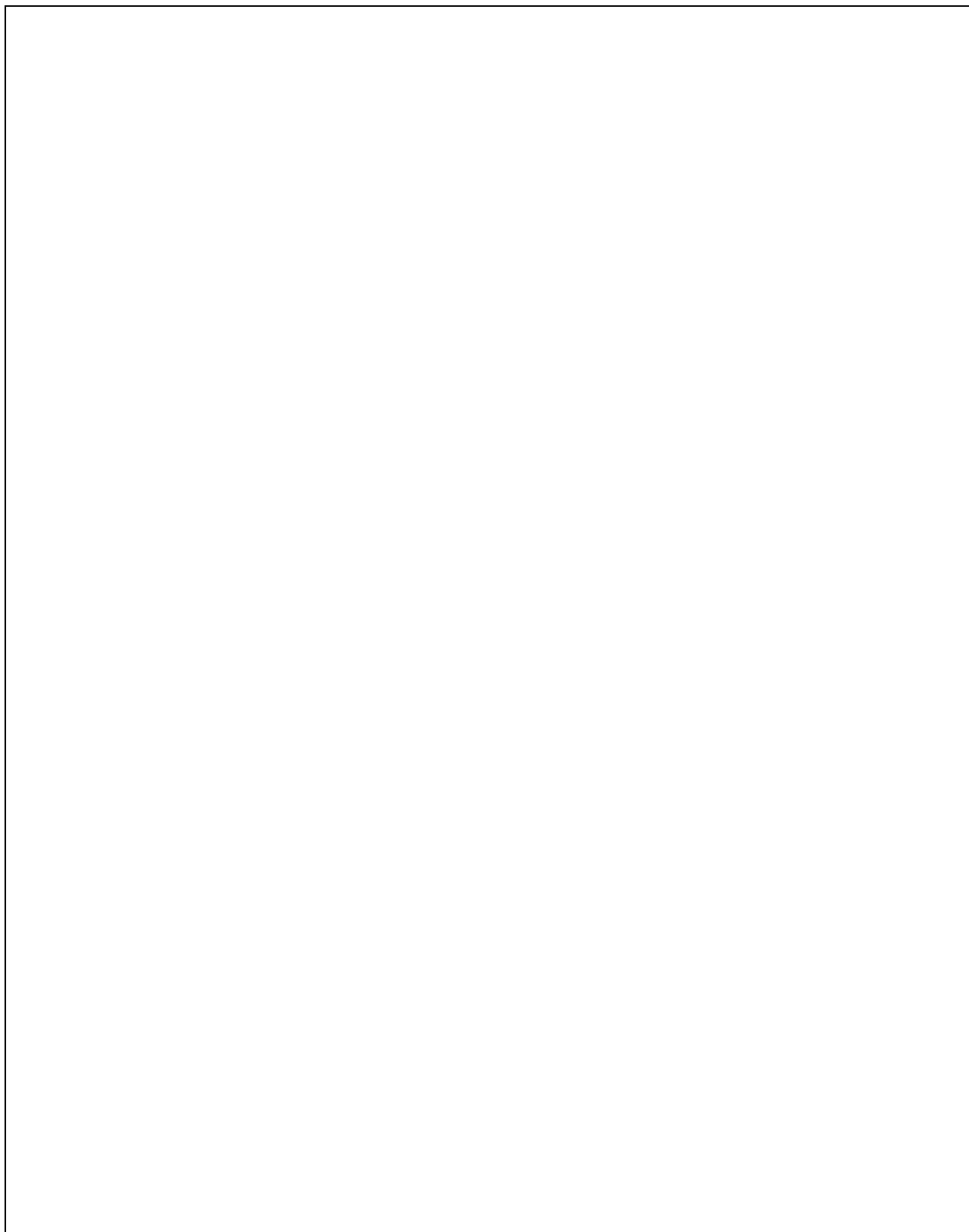
- b- A method Boolean find(String t) that tests whether a string *t* is contained in a sentence:

```
Sentence s = new Sentence("Mississippi!");  
  
boolean b = s.find("sip"); // returns true
```

- c- Use recursion to implement a method int indexOf(String t) that returns the starting position of the first substring of the text that matches t. Return -1 if t is not a substring of s. For example:

```
Sentence s = new Sentence("Mississippi!");  
  
int n = s.indexOf("sip"); // returns 6
```

```
public class Sentence  
{
```



Question 3 [15 points]

a) Consider the mergeSort program as covered in the lecture:

```
public static void mergeSort (Comparable[] a)
{
    if (a.length == 1)    // base case
        return;
    else
    {
        Comparable[] left = new Comparable[a.length/2];
        System.arraycopy(a,0,left,0,left.length);
        Comparable[] right = new Comparable[a.length-left.length];
        System.arraycopy(a, left.length, right, 0, right.length);

        mergeSort(left);
        mergeSort(right);

        merge(left, right, a);
    }
}
```

Explain in detail what the last instruction **merge(left, right, a)** does.

b) Consider the code of **merge** method:

```
1 public static void merge(Comparable[] left, Comparable[] right, Comparable[] whole)
2 {
3     int leftIndex = 0, rightIndex = 0, wholeIndex = 0;
4     while (leftIndex < left.length && rightIndex < right.length)
5     {
6         if(left[leftIndex].compareTo(right[rightIndex]) < 0)
7         {
8             whole[wholeIndex] = left[leftIndex];
9             leftIndex++;
10        }
11        else
12        {
13            whole[wholeIndex] = right[rightIndex];
14            rightIndex++;
15        }
16        wholeIndex++;
17    }
18    Comparable[] rest;
19    int restIx;
20    if (leftIndex >= left.length)
21    {
22        rest = right;
23        restIx = rightIndex;
24    }
25    else
26    {
27        rest = left;
28        restIx = leftIndex;
29    }
30
31    for (int i=restIx; i<rest.length; i++)
32    {
33        whole[wholeIndex] = rest[i];
34        wholeIndex++;
35    }
36 }
```

What is the role of instructions from 18 to 35? Justify in details your answer

Question 4 [20 points]

Write an applet which has two timers as shown below. The initial value of Timer 1 is 10, while the initial value of Timer 2 is 15. Both timers are represented by a JLabel. After a pause of exactly **one second**, **both timers decrease** their value by 1.

When Timer 1 reaches 0, the text is changed to "Timer 1: STOPPED" as shown below:

Initial Configuration	After Timer 01 reaches zero.

Similarly Timer 2 also displays "Timer 2: STOPPED" after reaching a value of zero. Each timer is represented by a JLabel. The height of the window is 100, and its width is 300 pixels.

Write an applet which implements the above timers using threads, as follows.

1) [2 points] Fill in the correct class and interface names below:

```
public class ThreadQuestion extends _____  
                                implements _____
```

2) [3 points] Declare the variables here corresponding to the timer values and the GUI components used for the timers:

- 3) [5 points] Write the code corresponding to the `init()` method of the applet (for initializing the applet)

```
public void init() {
```

}

- 4) [5 points] Write the code corresponding to the paint() method (for updating the applet)

```
public void paint(Graphics g) {
```

}

- 5) [5 points] Write the code corresponding to the run() method of thread (for decrementing the timers)

```
public void run() {
```

Question 5 [25 points]

- a) Write a static method ***listIntersection*** which takes as input two LinkedLists of Strings ***list1*** and ***list2*** and returns a LinkedList of Strings composed of the intersection of ***list1*** and ***list2***. That is, the returned LinkedList should contain only the Strings which are both in ***list1*** and ***list2*** and should not contain duplicates.

For example, if ***list1*** contains:

"ICS"	"201"	"KFUPM"	"ITC"	"SWE"	"Hello"	"ITC"	"201"
-------	-------	---------	-------	-------	---------	-------	-------

and ***list2*** contains:

"CCSE"	"SWE"	"102"	"ICS"	"CCSE"	"201"
--------	-------	-------	-------	--------	-------

then, ***listIntersection(list1,list2)*** should return the LinkedList:

"ICS"	"201"	"SWE"
-------	-------	-------

- b) Write a static method ***listUnion*** which takes as input two LinkedLists of Strings ***list1*** and ***list2*** and returns a LinkedList of Strings composed of the union of ***list1*** and ***list2***. That is, the returned LinkedList should contain the Strings which are either in ***list1*** or in ***list2*** and should not contain duplicates.

For example, if *list1* contains:

"ICS"	"201"	"KFUPM"	"ITC"	"SWE"	"Hello"	"ITC"	"201"
-------	-------	---------	-------	-------	---------	-------	-------

and *list2* contains:

“CCSE”	“SWE”	“102”	“ICS”	“CCSE”	“201”
--------	-------	-------	-------	--------	-------

then, *listUnion(list1,list2)* should return the LinkedList:

"ICS"	"201"	"KFUPM"	"ITC"	"SWE"	"Hello"	"102"	"CCSE"
-------	-------	---------	-------	-------	---------	-------	--------