



King Fahd University of Petroleum & Minerals
College of Computer Science and Engineering
Information and Computer Science Department
Second Semester 092 (2009/2010)

ICS 201 - Introduction to Computing II

Final Exam
Monday June 14th 2010
Time: 150 minutes

Name:

ID#:

Please circle your section number below:

<i>Section</i>	01	02	03	04
<i>Instructor</i>	Sami	Tarek	Sukairi	Sukairi
<i>Day and Time</i>	SMW 7 - 7:50	SMW 8 - 8:50	SMW 9 - 9:50	SMW 13:00 - 13:50

Question #	Maximum	Obtained
1	30	
2	35	
3	15	
4	20	
Total	100	

Question 1 [30 Points]

1- **[2 Points]** - Suppose we are sorting an array of eight integers using quick sort, and we have just finished the first partitioning with the array looking like this:

2 5 1 7 9 12 11 10

Which statement is correct?

- a. The pivot could be either 7 or 9.
- b. The pivot could be 7, but it is not 9.
- c. The pivot is not 7, but it could be 9.
- d. Neither 7 nor 9 is the pivot.

2- **[2 Points]**- Consider the following pseudocode:

```
for (k = 0 ; k < array.length - 1; k++) {  
    select the minimum element among  
        array[k]...array[array.length - 1];  
    swap the selected minimum with x[k];  
}
```

Which sorting algorithm is it?

- a. Selection Sort
- b. Insertion Sort
- c. Quick Sort
- d. Merge Sort

3- **[2 Points]** - Which of the following statements is true:

- a. Linear Search cannot be used on unsorted arrays.
- b. Binary Search cannot be used on unsorted arrays.
- c. Linear Search cannot be used on sorted arrays.
- d. Binary Search cannot be used on sorted arrays.

4- **[2 Points]**- Consider the following array of sorted integers

0	1	2	3	4	5	6	7	8
2	5	8	10	23	35	44	60	61

The values in the first row are the indices. How many iterations are needed to search for 35 using iterative binary search method?

- a. 1
- b. 2
- c. 3
- d. 4

5- **[2 Points]** - Which of the following statements about searching in a sorted array is not true?

- a. The linear search examines all values in an array until it finds a match or until it reaches the end.
- b. A binary search is generally more efficient than a linear search.
- c. A linear search generally takes more comparisons than a binary search.
- d. A binary search is always faster than a sequential search.

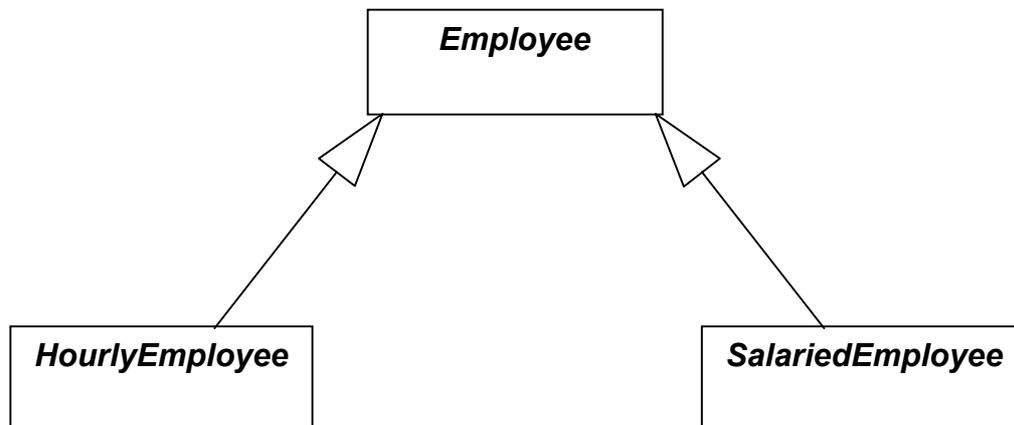
6- **[10 Points]** - Write a Java program that will read a sequence of positive real numbers entered by the user and will print the same numbers in sorted order from smallest to largest. The user will input a zero to mark the end of the input. Assume that at most 100 positive numbers will be entered.

8- **[10 Points]** - Use recursion to write a method `sumIntegers` which takes as input an integer value `n` and returns the sum of all natural numbers from 0 to `n`.

For example, if `n` is 6 the method should return 21 ($6+5+4+3+2+1$).

Question 2 [35 Points]

a) [15 Points] consider the following inheritance hierarchy:



An **Employee** has a *name* (*String*) and an *ID* (*int*). **SalariedEmployee** has an extra data member called *annualSalary* (*double*) for the annual salary. An **HourlyEmployee** has two extra data members: *wageRate* (*double*) for the payment he gets per hour of work and *hours* (*double*) for the number of hours he works per month.

Give the definition of classes: **Employee**, **SalariedEmployee** and **HourlyEmployee**. Every class should come with

- a constructor that initializes all instance variables
- a method *getMonthlySalary* that returns the monthly Salary.

Consider the following class *Product*:

```
public class Product {
    private String name;
    private double unitPrice;

    public Product(String n, double u)
    {
        name = n;
        unitPrice = u;
    }

    public double getUnitPrice()
    {    return unitPrice;    }

    public double getPrice(double quantity)
    {
        if(quantity > 0)
        {
            return (quantity*unitPrice);
        }else
        {
            System.out.println("Invalid quantity");
            return 0;
        }
    }
}
```

b) [5 Points] Write a class ***Company*** with three data members:

1. *name* (*String*)
2. a set of *Employees* (*TreeSet*)
3. a list of *Products* (*ArrayList*)

and one constructor *Company(String n)* that takes as input the name the *Company* and which initializes the set of *Employees* to the empty set and the list of *Products* to an empty list (0 elements).

c) [5 Points] Write a method *addEmployee* in the class ***Company*** to add an *Employee*. If the *Employee* is already in the *Company*, the method should throw an *Exception* and print in the catch block: "**Employee already in the Company**".

Assume that the file *Company.java* contains also the definition of the following inner class **Transaction** which represents a transaction done by an **Employee** to sell a **Product**:

```
public class Transaction {
    private Employee emp;
    private Product prod;

    public Transaction(Employee e, Product prod) {
        emp = new Employee(emp);
        prod = new Product(prod);
    }
    public Employee getEmployee()
    {    return new Employee(emp);    }
    public Product getProduct()
    {    return new Product(prod);    }
}
```

And assume that the class *Company* has a fourth data member which is a *LinkedList* of *Transactions*.

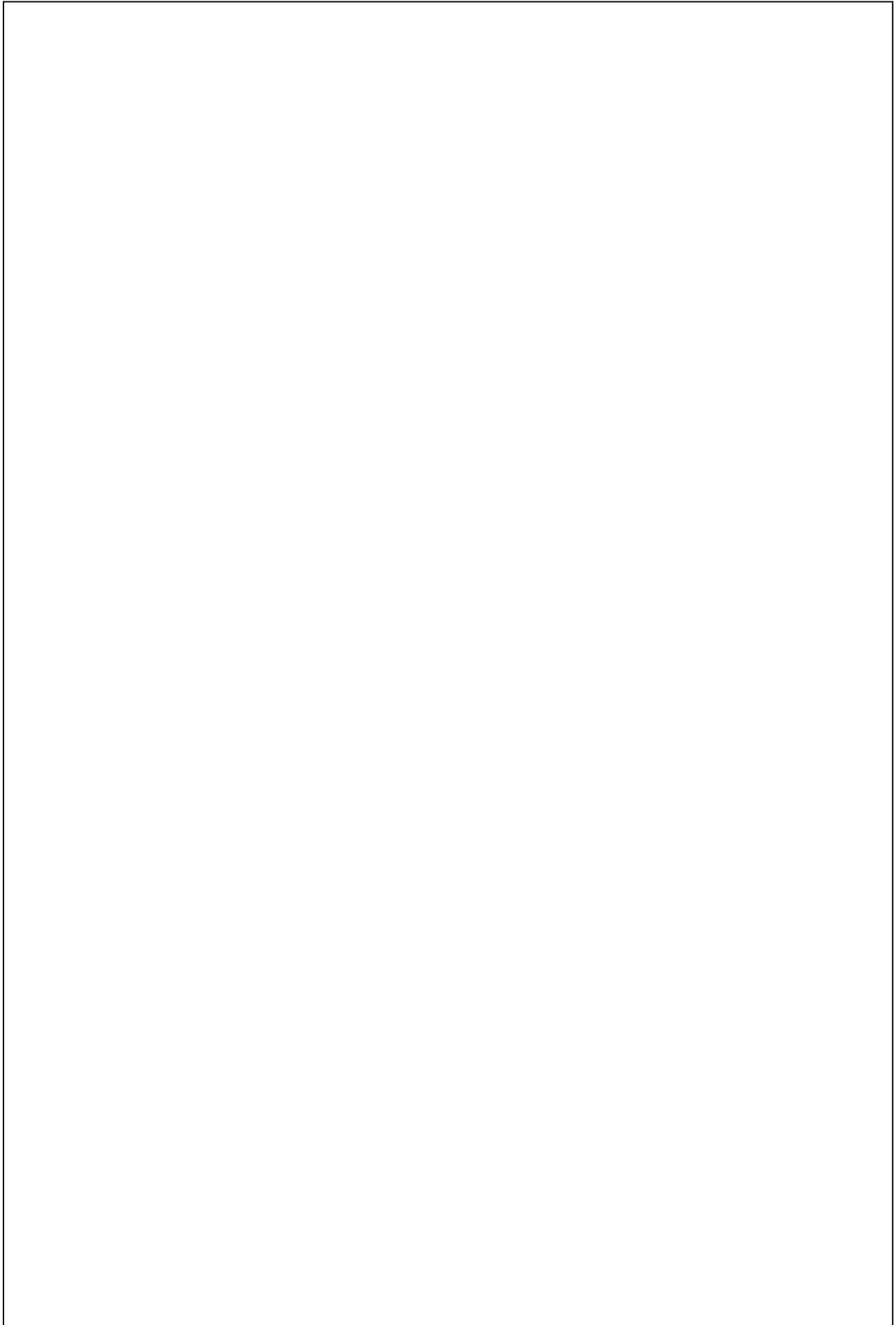
LinkedList<Transaction> transactions;

f) [5 Points] Write a method *activeEmployees* in class **Company** which returns the set (*TreeSet*) of all *Employees* that sold at least one *Product*.

g) [5 Points] Write a method *veryActiveHourlyEmployees* in class **Company** which returns the set (*TreeSet*) of all *HourlyEmployees* that sold at least one *Product* whose *unitPrice* is more than 100.

Question 3 [15 Points]

Write a GUI program that simulates a guessing game. Generate a random number between 1 and 100; that number is hidden from the user. Ask the user for a number between 1 and 100 in a text field, then tell the user whether the number is too high, too low, or the correct number. Let the user continue guessing until he guesses the correct number.



Question 4 [5+5+10 = 20 Points]

Write a *Garage* class with one instance variable: an *ArrayList* of Cars. Assume *Car* class is already defined and has the following API:

Car class API:**public class Car implements Comparable**

(comment: the natural ordering is based on the Maker data member)

Car()

Car(String maker, int model, int kilometerDriven)

Car(Car other)

String getMaker()

void setMaker(String MakerVal)

int getModel()

void setModel(int ModelVal)

int getKilometerDriven()

void setKilometerDriven(int kilometerDrivenVal)

String toString()

boolean equals(Object other)

Object clone()

int compareTo(Object other)

- a) a method returning the average number of kilometers of all cars in the garage

- b) a method returning all 2010 model cars in the garage.

- c) a method that displays all "Toyota" cars sorted by kilometer driven (in increasing order).

