UNIVERSITY OF OSLO

Candidate no

Faculty of mathematics and natural sciences

Examination in	INF1000 — Introduction to object-oriented programming
Day of examination:	Monday November 30th 2009
Examination hours:	14.30 – 17.30
This problem set consists of 20 pages.	
Appendices:	None
Permitted aids:	Any written or printed material

Please make sure that your copy of the problem set is complete before you attempt to answer anything.

- Read carefully through each problem before you try to solve it. Each problem is marked with a maximal score given for an entirely correct solution. The total max score is 100 points, which means that 5 points correspond to approximately 9 minutes and 10 points correspond to approximately 18 min (should you plan to come through all questions). Try to use your time well!
- Check that your examination set contains all pages before you answer any questions. If you miss any information in the formulation of an exercise, you may make your own assumptions as long as they are in accordance with the "spirit" of the problem. Such assumptions must be explained.
- You have to write your answers directly into these pages, not on separate sheets of paper. This applies to all questions, both multiple choice questions and questions where you are asked to write program code.
- In the questions where you are asked to write program code, we recommend that you first make a draft on a separate sheet of paper and then write your answer on this examination form at the appropriate place.
- Some of the questions are multiple choice questions. For these questions your score depends on how many correct answers you give. No score is given if you leave a question unanswered, or if you mark both answer alternatives.
- If you have marked an alternative with a cross and you later determine that you do not want a cross there, you can write "ERROR" immediately to the left of the box.
- You must write hard enough to make all copies that you hand in readable. *There must be no other papers from the examination text underneath when you write!*

(Continued on page 2.)

Contents

Good luck!

(Continued on page 3.)

Problem 1 Loops (5 points)

What value does the **int**-variable `number` have after each of the loops below?

1a

```
int number = 0;
for (int i=0; i<10; i++) {
    number++;
}
```

Svar: ______________________________

1b

```
int number = 0;
while (number < 10) {
    for (int i=10; i>0; i--) {
        number = number + 2;
    }
}
```

Svar: ______________________________

1c

```
int number = 0;
do {
    for (int i=0; i<20; i++) {
        number++;
    }
} while (number < 10)
```

Svar: ______________________________

1d

```
int number = 0;
for (int i=5; i>0; i--) {
    for (int j=i; j>0; j--) {
        number--;
    }
}
```

Svar: ______________________________

(Continued on page 4.)

Problem 2 Arrays and loops (5 points)

Assume that we have decleared a **boolean** array `yesno`, and that all positions in the array has received **boolean** values. Assume further that we have decleared an **int** variable `numberOfTrue`. The loops below attempt to count the number of positions in `yesno` that have the value **true** by means of the variable `numberOfTrue`. Are the loops correct?

Yes **No**

☐ ☐

```
numberOfTrue = 0;
for (int i=0; i<yesno.length; i++) {
    if (yesno[i]) {
        numberOfTrue++;
    }
}
```

☐ ☐

```
numberOfTrue = yesno.length;
for (int i=yesno.length; i>0; i--) {
    if (!yesno[i-1]) {
        numberOfTrue--;
    }
}
```

☐ ☐

```
numberOfTrue = 1;
for (int i=1; i<yesno.length; i++) {
    if (yesno[i-1]) {
        numberOfTrue = numberOfTrue+i;
    }
}
```

☐ ☐

```
numberOfTrue = 0;
for (int i=0; i<yesno.length; i=i+2) {
    if (yesno[i]) {
        numberOfTrue++;
    }
}
```

☐ ☐

```
numberOfTrue = -1;
for (int i=-1; i+1<yesno.length; i++) {
    if (yesno[i+1]) {
        numberOfTrue++;
    }
}
numberOfTrue++;
```

(Continued on page 5.)

Problem 3 Java syntax (5 points)

Are these expressions legal program statements in Java?

Yes	No	
☐	☐	`int i = (int) 1000;`
☐	☐	`int j = 'c';`
☐	☐	`boolean verdi = "INF1000";`
☐	☐	`String kurs = "INF1000";`
☐	☐	`int k = -1.0;`
☐	☐	`float f = -1.0F;`
☐	☐	`String[] tekst = { 1, 2, 3 };`
☐	☐	`String bokstav = 'a';`
☐	☐	`return "abc".equals("bcd");`
☐	☐	`boolean resultat = (resultat = true) == false;`

Problem 4 String (5 points)

Assume that the following variables have been declared:

```
String a = "kurator";
String b = "kur";
String c = "ator";
```

What is the value of the following **boolean** expressions?

true	false	
☐	☐	`a == (b+c)`
☐	☐	`a.equals(b+c)`
☐	☐	`a.startsWith(b)`
☐	☐	`a.compareTo(b) < 0`
☐	☐	`a.indexOf(c) == 4`

(Continued on page 6.)

Problem 5 Methods (10 points)

5a

A method `sum(int[] array)` is decleared below which is supposed to compute the sum of the numbers in an **int**-array. Will the method compile? If no, explain how one can modify the method to make it compile.

```
public void sum(int[] array) {
    int sum = 0;
    for (int i=0; i<array.length: i++) {
        sum += array{i};
    }
    return sum;
}
```

Write your answer here:

5b

Declear a method `average(int[] array)`, which computes the average value of the numbers in the parameter `array` of type **int**-array. The average value shall be computed with floating point division and the return value shall be of type **double**. You shall write code for the *whole* method, including the signature.

(Continued on page 7.)

5c

We can compute the average velocity of an object by dividing distance over time. Declare a method that takes two **double**-parameters `distanse` and `tid`, og which returns velocity as a **double**-value. If the value of `tid` is `0`, the method shall return `-1`. You shall write code for the *whole* method, included the signature. Find a suitable name for the method.

5d

In sea and air traffic the velocity is often given in *knots.* One knot is defined as one nautical mile pr. hour, which corresponds to 1852 meters pr. hour. Declare a method which takes a **double**-parameter `knots` and returns the corresponding velocity in *km/h* as a **double**-value. You shall write code for the *whole* method, including the signature. Find a suitable name for the method.

(Continued on page 8.)

Problem 6 UML (10 points)

Professor Smart is a regular customer at the restaurant “The Hungry Academic”. One day Prof. Smart is asked to help the restaurant design a Java program to keep track of *menus*, *dishes*, *ingredients* og *chefs*. Since Prof. Smart is only a Java beginner, he asks you to help him.

A menu has exactly five dishes, and a dish may occur in several menus. Each dish can contain at most five ingredients. Each menu has one responsible chef, and each ingredient has one chef that is responsible for purchasing sufficient supplies of the ingredient. One chef may be responsible for more than one menu and more than one ingredient.

Draw a UML class diagram for the Java program. Give names to the classes, draw association links between them and insert correct numbers on both sides of the association links. You do *not* have to insert variable and method names into the classes in the diagram.

Draw your UML diagram here:

(Continued on page 9.)

Problem 7 HashMap, objects and methods (25 points)

Professor Smart has started to write a Java program for "The Hungry Academic". He has declared a class `Chef`, listed below.

```java
public class Chef {

    private String name;            // chef's name
    private int phoneNr;            // chef's phoneNr
    private int yearOfEmployment; // year of chef's employment

    public Chef(String name, int phoneNr, int year) {
        this.name = name;
        this.phoneNr = phoneNr;
        this.yearOfEmployment = year;
    }

    public String toString() {
        return "Chef: " + name + ", tlf " + phoneNr +
            ", employed " + yearOfEmployment;
    }

    public String getName() {
        return name;
    }

    public int getPhoneNr() {
        return phoneNr;
    }

    public int getEmploymentYear() {
        return yearOfEmployment;
    }

    // Returns number of years the chef has been employed
    public int seniority() {
        Calendar cal = Calendar.getInstance();
        return cal.get(Calendar.YEAR) - yearOfEmployment;
    }

}
```

You don't have to understand how the method `seniority()` works. It is sufficient that you know that it returns the number of years a chef has been employed. You may freely use this method in your answers below.

(Continued on page 10.)

7a HashMap and loops (5 of 25 points)

Assume that Prof. Smart has declared

```
HashMap<String, Chef> chefs = new HashMap<String, Chef>();
```

and inserted some `Chef` objects in it. For each of the loops below, tick *one* of the options:

(a) it does *not* compile, or

(b) it compiles and prints out *all* the `Chef` objects in `chefs`,

(c) it compiles but it fails to print out all the `Chef` objects in `chefs`.

(a)	(b)	(c)	

☐	☐	☐	
			```
			for (Chef chef : chefs) {
			    System.out.println(chef.toString());
			}
			```

☐	☐	☐	
			```
			for (Chef chef : chefs.keySet()) {
			    String txt = chef.toString();
			    System.out.print(txt + "\n");
			}
			```

☐	☐	☐	
			```
			for (Chef chef : chefs.values()) {
			    System.out.println(chef.toString());
			}
			```

☐	☐	☐	
			```
			Iterator<Chef> iterator =
			    chefs.values().iterator();
			String tmp;
			while (iterator.hasNext()) {
			    tmp = iterator.next().toString();
			    System.out.println(tmp);
			    if (iterator.hasNext()) {
			        iterator.next();
			    }
			}
			```

☐	☐	☐	
			```
			Iterator<String> keyIterator =
			    chefs.keySet().iterator();
			Chef chef;
			while (keyIterator.hasNext()) {
			    chef = chefs.get(keyIterator.next());
			    System.out.println(chef.toString());
			}
			```

(Continued on page 11.)

7b Object variables (2 of 25 points)

The following code does not complile. How would you modify the code to make it compile?

```
Chef chef = new Chef("Eivind", 12345678, 1982);
chef.yearOfEmployment = 1983;
```

Write your answer here:

7c Object methods and HashMap (18 of 25 points)

Professor Smart declares the class `Restaurant` as shown below. The class contains for the time being only the declaration of the `HashMap` variable `chefs`, so Prof. Smart calls for your help to declare the following methods in the class. You may assume that the name of the chef shall be used as key in the `HashMap chefs`.

- **public boolean** `insertChef(Chef chef)`
  The method takes a `Chef` object as parameter. If there is a `Chef` object in `chefs` with the same name as `chef`, the method shall return **false** *without* adding `chef` to `chefs`. If, on the contrary, there is *no* `Chef` object in `chefs` with the same name as `chef`, `chef` shall be inserted into `chefs` and **true** shall be returned.

- **public boolean** `removeChef(String name)`
  This method shall remove the chef with the name `name` from `chefs`. The method shall return **true** if a `Chef` object was removed, and **false** if no chef with the name `name` was found.

- **public** `Chef findChef(String name)`
  This method shall check if there is a `Chef` object with name `name` in `chefs`. If there is such an object, this object shall be returned. Otherwise, **null** shall be returned.

(Continued on page 12.)

- **public int** `numberOfChefs()`
  This method shall return the number of `Chef` objects in `chefs`.

- **public** `Chef[] longestSeniority()`
  The method shall find the chef(s) that have been employed for the longest period of time (counted in years). The matching `Chef` objects shall be returned as a `Chef` array. If `chefs` is empty, the method returns an empty `Chef` array. If one chef alone has the longest seniority, the returned `Chef` array shall have length 1 and contain the respectice `Chef` object in position 0. If there are $n > 1$ chefs which share in having longest seniority, the `Chef` array to be returned has length n and contains *all* the respective `Chef` objects. You may use the method `seniority()` in class `Chef` for computing how many years a chef has been employed.

The methods shall be declared inside class `Restaurant` below.

```
public class Restaurant {

    HashMap<String,Chef> chefs = new HashMap<String,Chef>();

    // Deklare your methods here:
```

(Continued on page 13.)

(Continued on page 14.)

```
}
```

(Continued on page 15.)

Problem 8 Classes and objects (25 points)

Professor Smart shall complete the program for “The Hungry Academic”. He rapidly codes class `Ingredient` below.

```
public class Ingredient {

    private String name;
    private Chef responsibleChef;

    public Ingredient(String name, Chef chef) {
        this.name = name;
        this.responsibleChef = chef;
    }

    public String getName() {
        return name;
    }

    public Chef getResponsibleChef() {
        return responsibleChef;
    }
}
```

8a Data structure for ingredients in class `Dish` (10 of 25 points)

From the program specification in problem 6 it is clear that a dish may have maximum five ingredients. This shall be reflected in the program by class `Dish` having an `Ingredient` array with pointers to the `Ingredient` objects that the dish contains. Observe that a dish may have *less than* five ingredients. This must be accommodated. Finish class `Dish` below by inserting the follwing. The space in the forms should be sufficiently large for your answer.

1. Declare an `Ingredient` array with suitable number of positions and other variables that you may need for your solution.
2. Finish the method **public boolean** `insertIngredient(Ingredient ingredient)`. If a dish already contains five ingredients, the method shall return **false**. If the dish contains fewer than five ingredients, the method shall add `ingredient` to the next free position in the array that you declared in point 1 and return **true**.
3. Finish the method **public boolean** `containsIngredient(String name)`. The method shall search through all ingredients that the dish contains. If it finds an ingredient that is textually identical to `name`, the method shall return **true**. If such an ingredient is not found, the method shall return **false**.

(Continued on page 16.)

```
public class Dish {

    private String name;

    public Dish(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    // Declare the Ingredient array and other variables
    // you may need here:







    public boolean addIngredient(Ingredient ingredient) {

















    }
```

(Continued on page 17.)

```
    public boolean containsIngredient(String name) {








    }
}
```

8b How frequent are the ingredients used? (15 of 25 points)

Professor Smart has with your help almost completed the Java program. He has added the following declarations in class `Restaurant`:

```
    HashMap<String, Ingredient> ingredients =
        new HashMap<String, Ingredient>();
    HashMap<String, Dish> dishes = new HashMap<String, Dish>();
```

The professor has inserted test data into the system. As an extra bonus to the restaurant, he wishes to add a method in the `Restaurent` class which finds the number of times that each ingredient has been used in the dishes. Write a method that, for each ingredient, counts the number of dishes in which the ingredient has been used. The method shall print a list to the screen in which a line contains the name of an ingredient and the number of dishes in which the ingredient has been used. You may assume that each dish has a unique name.

Write the method on the next page.

(Continued on page 18.)

(Continued on page 19.)

Problem 9 The Norwegian law of privacy and data protection (Personopplysningsloven) (10 points)

Restaurant “The Hungry Academic” wishes to increase revenue by collecting information they already have from the customers: name, credit card number, what menus they have ordered, and the price of their orders. The restaurant extends this information with information they find on the Internet, like address, professional occupation, and information about tax from public taxation lists.

The restaurant has then planned to produce a “two–on–top” of the month by giving a free meal to both the customer that, within the last month, as eaten the most with respect to taxable income, and the customer that has spent most money in the last month at “The Hungry Academic”. The names of the winners will be published at the home page of the restaurant and contacted directly as well.

Is the restaurant allowed to freely store, use and publish such data? Explain in short terms how the “Personopplysningsloven” regulates this kind of use of person related data. Point to the relevant paragraphs that should be taken into consideration, and explain why they are relevant in this context.

(Continued on page 20.)