NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

Practical Examination for Semester 1, AY2008/9
CS1101 — Programming Methodology

8 November 2008 Time Allowed: 3 hours

INSTRUCTION TO CANDIDATES

1. This is an **OPEN book** examination. You are allowed to bring to the lab hard copies of notes and books. However, you are **NOT** allowed to bring into the lab digital/communication devices (such as laptop, cell phone, PDA, etc.) and storage devices (hard disk, thumb drive, CD, etc.). There are **2** exercises in **10** printed pages. The marking scheme for each task is as given – you should exercise the usual good programming practices (comments, proper indentation, meaningful naming convention, good design of classes etc.).
2. **Login to PC.** Please use the given ID (**cs1101**) and password (**to be revealed on PE day**) to log into your assigned PC with domain as **Computing**. Your NUSNET id will not be used as there is no connection to NUSNET. The PC comes with our recommended IDE **DrJava**. Java **manuals** are available on your desktop. We will not provide software/editor that are not already in the PC and we do not guarantee the availability of such software/editor. Please login to the CourseMarker (see paragraph 3) before invoking DrJava.
3. **Login to CourseMarker.** Click on the CourseMarker shortcut **c:\cmc\cs1101pe.bat** to log into the CourseMarker. Please use the **newly assigned password** (given to you just before the practical exam) to perform the login. Once you have gained access to the CourseMarker, please click on the setup button to download the necessary files and some dummy files – please **do not** delete any of these files though you are not using them in your programming, you need them when submitting your work to the CourseMarker. With this, you are ready to start programming.
4. **Working Directory.** Please create your files in the directory **"c:\Documents and Settings\cs1101\My Documents\CMhome\studentArea\yourID\exerciseDIR"** (where **yourID** is your CourseMarker login id and **exerciseDIR** is cs1101labPEex1, cs1101labPEex2) with the specified file names as stated in the questions. Only the **specified files** in the **specified directory** will be submitted to the CourseMarker when you click on the submit button of the CourseMarker client.
5. **Submission.** You are given **100** submissions. Note that unlike your lab, CourseMarker **does not** provide instant feedback for the PE**.** Only the **last version** will be graded. Note once again that the system **only** submits files of the specified filenames in the specified directory – all other files will not be captured by the CourseMarker. Please ensure you have submitted your work before the end of the examination – to avoid unnecessary stress to yourself, you should not wait till the very last minute of the exam to submit the work at the same time as many other students. The CourseMarker system will be closed shortly after the practical examination.
6. **Leaving the lab.** Please **log out** from the PC before you leave the lab. But please **DO NOT SHUT DOWN THE MACHINE.**
7. **No Communication.** You are NOT to communicate in any way (sharing of files, PCs, calculators, send emails, receive emails, use of ICQ, etc.) with anyone, other than the invigilators, during the practical examination. You must allow the invigilator access to your PC on request.
8. **Matriculation card.** Please put your matriculation card on the desk.

ALL THE BEST!

CS1101 AY2008/9 Semester 1
Practical Exam (PE)

Important notes

- This is an **Open-Book** examination.
- Time allowed: **3 hours**.
- There are **2** exercises in this PE. Exercise A constitutes 30%; Exercise B constitutes 70%.
- You are advised to spend some time thinking over the tasks to design your algorithms, instead of writing the programs right away.
- Manage your time well! Do not spend excessive time on any exercise or sub-task.
- Remember to submit your programs!

Please do not read the question until you are told to do so.

A. The True-False Grader (30%)

The results of a 10 true-or-false-question examination given to a computer science tutorial group are given in the following format where each line contains a student ID, followed by a single space, followed by a string of 10 characters containing his/her answers:

```
U081234X TTTFTFTTFT
U082081Y FFTFTTFFFT
U073984Z FTTFTFTTFT
U063846A FFTFTTTTTT
U045726B FFTFFFTTFF
U084635H TTFFTFFTFT
U098236F FFTFTFTTTT
U098363F TTTFTFTTFF
```

Write a program **TFGrader.java** that first reads in a string of 10 characters containing the correct answers. Next, the program reads in each student's data, and compares their answers with the correct answers. You should store the student ID and the number of correct answers (score) for each student.

The grade for each student is computed as follows:

1. Determine the highest score, *best*, in the group.
2. Let deviation be the difference between the highest score and the student's score, that is, *best* – score.
3. If the deviation is 0 or 1, award the grade A to student; if deviation is 2, award grade B; if the deviation is 3, award grade C. Otherwise, the student gets an F.

You should output the test results in three columns displaying the student ID, his/her score, and his/her grade.

Sample Run

A sample run is shown below. The outputs are shown in **bold**.

```
$ javac TFGrader.java
$ java TFGrader
Enter the correct answers:
TTTFTFTTFT
Enter the number of students:
8
Enter the student IDs and their answers:
U081234X TTTFTFTTFT
U082081Y FFTFTTFFFT
U073984Z FTTFTFTTFT
U063846A FFTFTTTTTT
U045726B FFTFFFTTFF
U084635H TTFFTFFTFT
U098236F FFTFTFTTTT
U098363F TTTFTFTTFF
The grades of the students are:
U081234X 10 A
U082081Y 5 F
```

```
U073984Z 9 A
U063846A 6 F
U045726B 6 F
U084635H 8 B
U098236F 7 C
U098363F 9 A
```

A template program **TFGrader.java** is provided below. It is also available in your working directory.

```
// TFGrader.java
// Author:
// Matriculation number:
// Lecture group (X, Y or Z):
// Discussion group (1-8):

/**
 * This is the definition of the TFGrader class.
 */

import java.util.*;

class TFGrader {

    public static void main( String[] args ) {
        Scanner stdIn = new Scanner(System.in);
        System.out.println("Enter the correct answers:");
        //fill in your code here

        System.out.println("Enter the number of students:");
        //fill in your code here

        System.out.println("Enter the student IDs and their answers:");
        //fill in your code here

        System.out.println("The grades of the students are:");
        //fill in your code here

    } // main

} // TFGrader
```

Marking Scheme: Total of 30 marks

1. Can the program be compiled? 5 marks
2. Use of array and correct processing of array? 5 marks
3. Style (including your particulars, use of constant, comments, spacing/indentation, choice of variable names): 6 marks
4. Correctness of result: 14 marks (7 test data sets; 2 marks for each data set.)
5. The marks for this task will constitute **30%** of the total marks.

B. The Fish Class and FishTank Class (70%)

Design a class named **Fish** to represent a fish. The class contains:

- A string data field named **species** that specifies the species of the fish (default guppy).
- A character data field named **gender** that specifies whether the fish is male ('m') or female ('f'). You may assume that the character is lowercase and that the default is 'm'.
- A double data field named **weight** that specifies the weight of the fish (default 1.0 gram).
- A zero-argument constructor that creates a default fish.
- A three-argument constructor that creates a fish with user-specified species, gender and weight.
- Accessor and mutator methods for all three data fields.
- A method named **feed()** that increases the weight of the fish by 10%.
- A method named **toString()** that returns a string description for the fish.

Design a class named **FishTank** to simulate a fish tank. The class contains:

- A class constant named **MAX_FISH** with value 200 to denote the maximum number of fish that the fish tank can hold.
- An array of fish objects **container[]** that are instances of the Fish class defined above.
- An int data field named **totalFish** that keeps track of the total number of fish in the tank.
- A method named **addFish(Fish f)** to add a new fish f to the tank.
- A method named **feedFish()** that increases the weight of all the fish by 10%.
- A method named **feedFish(String species)** that only increases the weight of fish of the specified species by 10%.
- A method named **getTotalFish()** that returns the total number of fishes.
- A method named **getFish(int index)** that returns the Fish at the specified index.
- A method named **reproduceFish()** that simulates the reproduction of fish in the tank. Each female fish in the tank will produce 6 new offsprings (3 males and 3 females) as long as there is 1 male fish of the same species.
  For example, suppose the fish tank has 3 female guppies, 1 female goldfish and 2 male guppies. Each female guppy will produce 6 new guppies, while the female goldfish will not reproduce. The tank will now have 18 new offsprings (9 new female guppies and 9 new male guppies), giving a total of 12 female guppies, 1 female goldfish and 11 male guppies.
- A method named **computeOccupancyRate()** that returns the **percentage** of the number of fish in the tank to the maximum number of fish the tank can hold. The percentage is of type double.

You may assume that, at no time, the number of fish in the fish tank will exceed MAX_FISH.

C. The FishTankDriver Class (0%)

In order to test your programs, we have included a completed driver program (FishTankDriver.java) which reads in commands and performs operations on your FishTank class. Note: Do NOT modify FishTankDriver.java. It will NOT be graded. If a command has arguments, then each of the arguments is separated by a single space. The commands are as follows:

addfish	This adds a default fish to your fish tank.
addfish <species> <gender> <weight>	This adds a fish with specified <species> <gender> and <weight>. <gender> can only take on TWO possible values 'f' or 'm'.
feedfish	This feeds all the fishes.
feedfish <species>	This feeds all fishes of specified species.
reproduce	This reproduces all the fishes as described in the writeup above.
occupancy	This prints the occupancy rate of the tank in percentage.
quit	This quits the program and prints out the program output. The first line of the output contains a single number *n* denoting the number of fishes. The next *n* lines contains the information of a fish each in the format <species> <gender> <weight>, each of the terms separated by a single space.

Sample Run

A sample run is shown below. The outputs are shown in **bold**.

```
$ javac FishTankDriver.java
$ java FishTankDriver
addfish
addfish guppy m 3
feedfish
occupancy
1.0%
quit
2
guppy m 1.1
guppy m 3.3000000000000003
```

The templates for the Fish and FishTank classes are given below. The completed FishTankDriver class is also given, and you should NOT change it.

Fish.java

```
// Fish.java
// Author:
// Matriculation number:
// Lecture group (X, Y or Z):
// Discussion group (1-8):

/**
 * Complete the definition of the Fish class.
 */


public class Fish {

    public Fish()
    {

    }

    public Fish(String species, char gender, double weight)
    {

    }

    public char getGender()
    {

    }

    public void setGender(char gender)
    {

    }

    public double getWeight()
    {

    }

    public void setWeight(double weight)
    {

    }

    public String getSpecies()
    {

    }

    public void setSpecies(String species)
    {

    }

    public void feed()
    {

    }

    public String toString()
    {

    }

} // Fish
```

FishTank.java

```
// FishTank.java
// Author:
// Matriculation number:
// Lecture group (X, Y or Z):
// Discussion group (1-8):

/**
 * Complete the definition of the FishTank class.
 */


public class FishTank {

    public FishTank()
    {

    }

    public void addFish(Fish f)
    {

    }

    //Feed all fishes
    public void feedFish()
    {

    }

    //Feed only specified species
    public void feedFish(String species)
    {

    }

    //Count the number of fishes
    public int getTotalFish()
    {

    }

    //Get fish at the specified index
    public Fish getFish(int index)
    {

    }

    //Returns the occupancy rate of the fish tank
    public double computeOccupancyRate()
    {

    }

    //Reproduce fishes based on citeria specified in document
    public void reproduceFish()
    {

    }

} // FishTank
```

FishTankDriver.java

```
// FishTankDriver.java
// Author:
// Matriculation number:
// Lecture group (X, Y or Z):
// Discussion group (1-8):

/**
 * This is the completed driver class for the FishTank class. DO NOT MODIFY IT.
 */

import java.util.*;

public class FishTankDriver {

    public static void main( String[] args ) {

    Scanner stdIn = new Scanner(System.in);
    String[] line = {};
    FishTank tank = new FishTank();
    String newLine = "";
    do {
      newLine = stdIn.nextLine();

      // We split up the input into several tokens by the spaces
      // The first item of the resultant array would be command
      // and subsequent items are the parameters.
      line = newLine.split(" ");

      //Add fish command
      if (line[0].equalsIgnoreCase("addfish"))
      {
        //Add default fish
        if(line.length == 1)
        {
          Fish f = new Fish();
          tank.addFish(f);
        }
        //Add user specified fish
        else
        {
          String species = line[1];
          char gender = line[2].charAt(0);
          double weight = Double.parseDouble(line[3]);
          Fish f = new Fish(species, gender, weight);
          tank.addFish(f);
        }
      }
      else if(line[0].equalsIgnoreCase("feedfish"))
      {
        //Feed all fishes
        if(line.length == 1)
        {
          tank.feedFish();
        }
        //Else feed only particular species
        else
        {
          String species = line[1];
          tank.feedFish(species);
        }
      }
      else if(line[0].equalsIgnoreCase("reproduce"))
      {
        tank.reproduceFish();
      }
      else if(line[0].equalsIgnoreCase("occupancy"))
      {
        System.out.println(tank.computeOccupancyRate() + “%”);
      }
```

```
    } while (!newLine.equals("quit"));

    int numFish = tank.getTotalFish();
    System.out.println(numFish);
    for(int i = 0; i < numFish; i++)
    {
      System.out.println(tank.getFish(i));
    }

  } // main

} // FishTankDriver
```

Marking Scheme: Total of 70 marks

1. Can the programs be compiled? 10 marks
2. Style (including your particulars, use of constant, comments, spacing/indentation, choice of variable names): 10 marks
3. Correctness of results: 50 marks (16 test data sets)
4. The marks for this task will constitute **70%** of the total marks.