SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

ICT 131
Introductory Programming and Object-Oriented Concepts Using Java

Tutor-Marked Assignment

July 2017 Presentation

TUTOR-MARKED ASSIGNMENT

This tutor-marked assignment is worth 21% of the final mark for ICT131 – Introductory Programming and Object-Oriented Concepts Using Java.

The cut-off date for this assignment is **Monday, 28 August 2017, 2355 hours.**

Submit your solution document in the form of a *single* MS Word file, clearly headed with your name and your personal identifier, on or before the cut-off date shown above. You should make only one submission for a TMA.

NOTE: You are to copy the text of each program you write into your solution document when asked to submit the program listing. If you submit the program listing as a screenshot, you will be heavily penalized. Submit screenshots for only the output of your program.

Marks will be allocated for tidiness so that your tutor will have minimum difficulties in assessing your answers. In particular, take note that program code should follow proper code convention in layout and format. This includes:

(a) The use of meaningful and appropriate variable names

(b) Proper indentation of program lines

(c) Differentiation between uppercase and lowercase especially in variable names

(d) Suitable but not excessive comments, if necessary

Question 1 [25 marks]

The objectives of this question are

(a) to declare variables

(b) to write simple Java program without selection and repetition structures

(c) to write Java program with selection structure

(a) Note: You are NOT allowed to use selection or repetition structure for Question 1(a).

Write a program that accepts **TWO (2)** 3-digit whole numbers from the command line, and output the difference between the first 3-digit number and the second 3-digit number (divisor). Assume that the first number is greater than the second number.

For example, if the two numbers are 359 and 178, the program should output:

```
  3 5 9
- 1 7 8
---------
  1 8 1
```

If the two numbers are 602 and 529, then the program should output:

```
  6 0 2
- 5 2 9
---------
  0 7 3
```

Note that

- **You should not hardcode the values of the integers to be used in your program. Instead, your program must read the values from the command line.**
- **Your program must compute the difference ONE digit at a time.**
- **You are not allowed to treat the value as a string and you cannot use charAt() method.**

Submit your program listing together with screenshots showing **THREE (3)** runs of your program with the sample inputs (“379” , “178”) , (“602” , “529”) on the command line and another pair of your own choice. Include your name and student number on the first line of the output.

(10 marks)

(b) To determine the cost of monthly travel pass, refer to Table Q1(b) from the webpage https://www.ptc.gov.sg/FactsAndFigures/fares.htm, reproduced here.

Bus Monthly Concession Passes (Unlimited bus rides)	
Cardholders	Price
Primary Student	$22.50
Secondary Student	$27.50
Polytechnic Student	$27.50
University Student	$52.00
Full-time National Serviceman	$52.00

Train Monthly Concession Passes (Unlimited train rides)	
Cardholders	Price
Primary Student	$20.00
Secondary Student	$25.00
Polytechnic Student	$25.00
University Student	$45.00
Full-time National Serviceman	$45.00

Hybrid Monthly Concession Passes (Unlimited bus and train rides)	
Cardholders	Price
Primary Student	$41.00
Secondary Student	$51.00
Polytechnic Student	$51.00
University Student	$85.00
Full-time National Serviceman	$85.00

Adult Monthly Travel Pass (Unlimited bus and train rides)	
Cardholder	Price
Adult	$120.00

Senior Citizen Monthly Concession Pass (Unlimited bus and train rides)	
Cardholder	Price
Senior Citizen	$60.00

Monthly Concession Pass for Persons with Disabilities (Unlimited bus and train rides.)	
Cardholder	Price
Persons with Disabilities	$60.00

Table Q1(b)

Outline the control structures of structured programming to write a program that prompts and reads the type of person the monthly pass is purchased for, and the type of pass **only if** the person type is a student or a fulltime national serviceman.

The program then uses Table Q1(b) to determine and display the cost of the monthly pass with 2 decimal places.

Note that

- that duplicate/repeated code will be penalised.
- You must use three selection structures: `if`, `if-else` and `switch` as well as the logical operators: `&&` and `||`.

FOUR (4) sample runs are shown below with user inputs are underlined for readability:

Run 1:

```
Choose the type of monthly pass
1  - Primary Student
2  - Seccondary Student
3  - Polytechnic Student
4  - University Student
5  - Full-Time National Serviceman
6  - Adult
7  - Senior Citizen
8  - Person with Disability
Your choice: 3
Enter the type of monthly pass
B - Bus
T - Train
H - Hybrid (Bus and Train)
Your choice: h
Cost of pass = $51.00
```

Run 2:

```
Choose the type of monthly pass
1  - Primary Student
2  - Seccondary Student
3  - Polytechnic Student
4  - University Student
5  - Full-Time National Serviceman
6  - Adult
7  - Senior Citizen
8  - Person with Disability
Your choice: 6
Cost of pass = $120.00
```

Run 3:

```
Choose the type of monthly pass
1  - Primary Student
2  - Seccondary Student
3  - Polytechnic Student
4  - University Student
5  - Full-Time National Serviceman
6  - Adult
7  - Senior Citizen
8  - Person with Disability
Your choice: 0
Invalid type of person
```

Run 4:

```
Choose the type of monthly pass
1  - Primary Student
2  - Seccondary Student
3  - Polytechnic Student
4  - University Student
5  - Full-Time National Serviceman
6  - Adult
7  - Senior Citizen
8  - Person with Disability
Your choice: 1
Enter the type of monthly pass
B - Bus
T - Train
H - Hybrid (Bus and Train)
Your choice: A
Invalid type of pass
```

Submit your program listing together with screenshot showing **FOUR (4)** runs of your program for varying output. Include your name and student number on the last line of the output.

(15 marks)

Question 2 [20 marks]

The objectives of this question are

(a) to practice the use of selection structure in simple Java program

(b) to practice the use of repetition structure in a Java program

(c) to practice the use of method in a Java program

(d) to manipulate string

Without using arrays or regular expressions, write a Java program to implement a spelling game using only strings, string buffers, characters and integers. Use the Scanner class to read user input and the Math class to generate random numbers.

For the game, adhere strictly to the following constraints:

- Use a string to store these 10 words.

```
String s = "meringue albumen foulard eudaemonic narcolepsy elucubrate"
           + " vivisepulture pococurante cymotrichous malfeasance";
```

- Use a Scanner object to read from this string.

```
Scanner scanString;
scanString = new Scanner(s);  // scanString starts from beginning of s
```

  For example, to read the second word `albumen` into a String variable `word`, use

```
scanString.next(); // move over first word
word = scanString.next();
```

- Use the Math class to generate a random number for a position of a word in `s`. The valid positions are 0 to 9, with 0 being the position of the first word `meringue` and 9 the position of the last word `malfeasance`. The player will guess the spelling of the randomly chosen word.

- The player guesses one letter a time, and can guess a maximum of 5 letters after which he must spell the whole word.
  - For each correctly guessed letter, show the letter in all its position in the word. Refer to the sample run.
  - If a player has guessed at least a correct letter and he has not exhausted all 5 letters guesses, prompt the user whether he wishes to spell the whole word. This means that the player can spell the whole word without having to complete 5 guesses of individual letters, as long he has guessed one letter correctly.

- At the end of a game, the program prompts the player whether he wants to spell a new word. The user should enter `n` if he wishes to exit the program.

- Outline the control structures of structured programming and use **ALL** three repetition structures: `for`, `while` and `do-while`.

Note that you must NOT use arrays or regular expressions.

An example run is shown here with user input underlined for readability:

```
Spell this 11-letter word in 5 tries
-----------
Guess a letter: i
Outcome: -----------
Guess a letter: b
Outcome: -----------
Guess a letter: o
Outcome: -o-o-------
Do you want spell the word now? (y/n): n
Guess a letter: a
Outcome: -o-o---a---
Do you want spell the word now? (y/n): n
Guess a letter: e
Outcome: -o-o---a--e
Spell the complete word: pococulante
You are incorrect.
The correct word is pococurante
Spell another word? (y/n): y
Spell this 7-letter word in 5 tries
-------
Guess a letter: a
Outcome: ----a--
Do you want spell the word now? (y/n): n
Guess a letter: e
Outcome: ----a--
Do you want spell the word now? (y/n): n
Guess a letter: o
Outcome: -o--a--
Do you want spell the word now? (y/n): n
Guess a letter: l
Outcome: -o-la--
Do you want spell the word now? (y/n): n
Guess a letter: d
Outcome: -o-la-d
Spell the complete word: foulard
You are correct!
The correct word is foulard
Spell another word? (y/n): y
Spell this 13-letter word in 5 tries
-------------
Guess a letter: v
Outcome: v-v----------
Do you want spell the word now? (y/n): n
Guess a letter: e
Outcome: v-v--e------e
Do you want spell the word now? (y/n): n
Guess a letter: u
Outcome: v-v--e-u--u-e
Do you want spell the word now? (y/n): y
Spell the complete word: vivisepulture
You are correct!
The correct word is vivisepulture
Spell another word? (y/n): n
Thank you for playing Spell The Word!
```

Submit your program listing together with screenshot showing **TWO (2)** runs of your program output. Include your name each time the correct word is revealed to the player.

(20 marks)

Question 3 [35 marks]

The objectives of this question are

(a) to practice arrays in Java.

(b) to practice methods in Java.

Develop a Java program using the control structures of structured programming on an array data structure for a Java staff leave management system.

Use several arrays to record the details of staff:

- an array `StaffNames` to record name of staff whose leave is being managed,
- an array `staffPositionCode` to record which type of position they are in (T – top management, M- management, S – supervisory and J – junior level)
- an array `staffNumleaves` to record the current leave balance of staff.

Use several arrays to record the details of positions and the annual leave entitlement.

- an array `positionCode` to record 4 types of positions (T – top management, M-management, S – supervisory and J – junior level)
- an array `numleaves` to record the annual leave entitlement according to Table Q3.

Position Code	Leave (days)
T	52
M	28
S	21
J	14

Table Q3

Write a complete Java program with the following methods:

(a) A method `search` to locate the staff in the array given the staff name.

The method accepts the `name` of the staff to be searched. In addition, it accepts the `StaffNames` array and an integer count which is the number of staff already recorded in the array.

If the `name` exists in `StaffNames`, the method returns the array index where the `name` is located. Otherwise, the method returns -1.

(3 marks)

(b) A method `addStaff` to record details of a staff.

The method accepts the `name` and `positionCode` of the staff to add. In addition, it accepts all the arrays and an integer count which is the number of staff already recorded in the arrays for staff details.

The method checks whether provided `name` already exists. If `name` exists, the method prints the error message: `Xxx is already a staff!` where `Xxx` is the provided name.

If `name` does not exist, the method records `name` and `positionCode` in the arrays `StaffNames` and `staffPositionCode` respectively, set the leave of the staff using both the `positionCode` and `numLeaves` arrays, increments `count` as one more staff is added and prints the message: `Successfully added Xxx with yy days of leaves!` where `Xxx` is the provided name and `yy` is the number of days of leave the staff has, based on his position code.

This method returns the value of `count`.

(7 marks)

(c) A method `removeStaff` to remove the staff given the staff name.

The method accepts the `name` of the staff to be removed. In addition, it accepts the arrays for the staff details and an integer count which is the number of staff already recorded in these arrays.

The method checks whether provided `name` exists. If `name` does not exist, the method prints the error message: `Can't remove Xxx: Reason: Not a staff` where `Xxx` is the provided name.

If `name` exists, the method removes the staff by shifting the details of other staff to pack the array, so that only array positions `0` to `count-2` store existing staff details. The method then prints the message: `Successfully removed: Xxx` where `Xxx` is the provided name.

This method returns `true` if the remove is successful and `false` otherwise.

(6 marks)

(d) A method `takeLeave` to allow a staff to take leave.

The method accepts the `name` of the staff taking leave, the number of days of leave to take (`numDays`), the arrays `staffNames` and `staffNumleaves`, and an integer `count` which is the number of staff already recorded in the arrays.

The method checks whether provided `name` exists. If `name` does not exist, the method prints the error message: `Can't apply leave Xxx Reason: Not a staff` where `Xxx` is the provided name. The method then returns an integer which is the largest number in the `Integer` class.

If `name` exists, the method checks whether the staff has sufficient number of leave. If there is insufficient leave, the method prints the error message: `Can't apply leave Xxx Reason: Insufficient leave` where `Xxx` is the provided name. The method then returns a negative number which is the number of days exceeded had the leave been taken.

If there is sufficient leave, the method updates the number of days of leave left for the staff and prints the message: `Successful leave application for Xxx` where `Xxx` is the provided name. The method then returns a number which is the number of days of leave left after the leave is taken.

(6 marks)

(e) A method `listStaffWithLeave` to display staff with non-zero days of leave. The method accepts the necessary arrays `staffNames` and `staffNumleaves`, and an integer `count` which is the number of staff already recorded in the arrays.

The method displays the header `List of Staff who can Take Leave` followed by 0 or more staff detail lines (name and days of leave left) and then the trailer `End of List`.

A sample output is shown below:

```
List of Staff who can Take Leave
Dawn Ho    14 days
Alan Tan   21 days
End of List
```

(3 marks)

(f) The method `displayMenu` which displays this menu:

```
Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
```

(2 marks)

(g) The method `main` which does the following actions:

- Declare and create the arrays `StaffNames`, `staffPositionCode`, `staffNumleaves`, `positionCode` and `numleaves`.

  For testing purposes, let `StaffNames`, `staffPositionCode` and `staffNumleaves` record a maximum of 2 staff data each.

  Initialise `positionCode` and `numleaves` according to Table Q(3).

- Declare and initialise `count`, the number of data already recorded in the staff detail arrays to zero,

- Repeatedly display a menu to allow the user to select an option and **call** the **appropriate method** to perform the task according to the selected option until the user chooses to exit the program.
  - For option 1, the method checks whether the count of data is already the maximum number of data that can be stored in the staff detail arrays. If so, print the error message: `No more staff can be added`. Otherwise, the method prompts and reads the name and position code of the staff to add.
  - For option 2, the method prompts and reads the name of the staff to be removed. If the remove is successful, decrement `count`.
  - For option 3, the method prompts and reads the name of the staff and the number of days of leave to take.

    If the leave is successfully taken, print the message: `Xxx has yy day(s) of leave left` where `Xxx` is the provided name and `yy` is the number of days of leave the staff has left.

    If the leave is not successfully taken, print two messages:
    - `Applying yy days will result in -zz days leave!` where `yy` is the number of leave the staff wishes to take and `zz` is the number of days exceeded had the leave been taken

- `Xxx can apply only yy day(s) of leave` where `Xxx` is the provided name and `yy` is the number of days of leave the staff has, and so can take.

(8 marks)

An example run is shown here with user input underlined for readability:

```
Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 1
Enter name of staff to add: Dawn Ho
Enter position code of staff to add (T, M, S or J): J
Successfully added Dawn Ho with 14 days of leave

Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 1
Enter name of staff to add: Alan Tan
Enter position code of staff to add (T, M, S or J): S
Successfully added Alan Tan with 21 days of leave

Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 4
List of Staff who can Take Leave
Dawn Ho    14 days
Alan Tan   21 days
End of List

Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 2
Enter name of staff to remove: Peter
Can't remove Peter: Reason: Not a staff

Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 2
Enter name of staff to remove: Dawn Ho
Successfully removed: Dawn Ho

Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 4
List of Staff who can Take Leave
Alan Tan   21 days
End of List

Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 3
Enter name of staff taking leave: Alan Tan
```

```
Enter number of days to apply leave for: 10
Successful leave application for Alan Tan
Alan Tan has 11 day(s) of leave left

Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 3
Enter name of staff taking leave: Alan Tan
Enter number of days to apply leave for: 14
Can't apply leave Alan Tan: Reason: Insufficient leave
Applying 14 days will result in -3 days leave!
Alan Tan can apply only 11 day(s) of leave

Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 3
Enter name of staff taking leave: Alan Tan
Enter number of days to apply leave for: 11
Successful leave application for Alan Tan
Alan Tan has 0 day(s) of leave left

Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 4
List of Staff who can Take Leave
End of List

Menu
1. Add Staff
2. Remove Staff
3. Take Leave
4. List Staff Leave Details
0. Exit
Enter option: 0
Closing application
```

Submit the program listing together with a screenshot showing the output. In your run, you should demonstrate that each of the options and repetitions are working. Include your name and student number in your program output whenever a method is called.

Question 4 [20 marks]

The objectives of this question are

(a) to write a basic Java class

(b) to write a simple tester to test your Java class

You have been engaged to **develop a prototype using the object-oriented programming approach** for the leave application and to implement classes for this object-oriented application.

(a) Write programs using object oriented programming approach to implement a Java class to model a Staff. This class has the following:

- Suitable instance variables for
  - Name of staff
  - Position code (T – top management, M- management, S – supervisory and J – junior level)
  - Current leave balance
  - Whether the staff is allowed to exchange leave for money

- Two constructors, one with the input values for all the instance variables and the other with values for all the instance variables except for: whether the staff is allowed to exchange leave for money (set to set to the default value false) and position code is set to J. Avoid repeating assignment statements by making one of the constructors invoke the other constructor.

- Get and set methods for current leave balance and the position code.

- The get method for whether the staff is allowed to exchange leave for money.

- A `changeIsAllowedMoneyExchange()` method that does not have any parameter and does not return any value. This method changes a staff who is not allowed to exchange leave to money to allowed, and changes a staff who is allowed to exchange leave to money to not allowed.

- A `hasLeave()` method which returns true if the current leave balance is more than 0. Returns false otherwise.

- A `computeLeaveToMoney(double)` method which returns amount of money the staff would receive should his current leave balance be converted to money. The method returns the amount by computing the product of the current leave balance and the input parameter which is the rate per day, only if the staff is allowed to make the exchange. Otherwise, the method returns -1.

- A `compareTo()` method that takes as parameter another staff, compares the **current leave balance** of this staff and the parameter, and returns 1 of this staff has more days of leave, 0 if their current leave balances are the same, and -1 if this staff has fewer days of leave.

- A `toString()` method that returns with descriptions, the values of the instance variables and the amount of money when current leave balance is converted to money, if the staff is allowed to exchange leave for money. Fix the rate per day to 112.50.

(14 marks)

(b) Write a test program to exercise the class written in part (a) above. The test program should perform the following:

- Create the following 2 staff using the appropriate constructors, **hold the objects in a single suitable data structure**.

Name	Position Code	Current Leave Balance	Allowed to Exhange
Dawn Ho	J	14	False
Alan Tan	S	21	True

Table Q4(g)

- Apply the `changeIsAllowedMoneyExchange ()` method to the first staff.
- Take away 21 days of leave from the second staff.
- Use a loop to print details of staff only if they have leave.

Submit your program listing together with screenshot of your output. Include your name and student number on the first line of the program output.

(6 marks)

----- END OF TMA -----