UNIVERSITI MALAYA
UNIVERSITY OF MALAYA

PEPERIKSAAN IJAZAH SARJANA MUDA SAINS KOMPUTER/SARJANA MUDA TEKNOLOGI MAKLUMAT
EXAMINATION FOR THE DEGREE OF BACHELOR OF COMPUTER SCIENCE/BACHELOR OF INFORMATION TECHNOLOGY

SESI AKADEMIK 2014/2015 : SEMESTER II
ACADEMIC SESSION 2014/2015 : SEMESTER II

WXES1116 : Pengaturcaraan I
Programming I

Jun 2015
June 2015

Masa: 2 jam
Time: 2 hours

ARAHAN KEPADA CALON:
INSTRUCTIONS TO CANDIDATES:

Calon dikehendaki menjawab **SEMUA** soalan (50 markah).
*Answer **ALL** questions (50 marks).*

(Kertas soalan ini mengandungi 4 soalan dalam 6 halaman yang dicetak)
(This question paper consists of 4 questions on 6 printed pages)

1. Modifikasi kod di dalam Q1.java supaya ianya akan memaparkan output "H3110 w0r1d 2.0". Isytiharkan pembolehubah-pembolehubah secara berasingan untuk menyimpan setiap yang berikut: **H**, **3**, **1**, **0**, **w**, **r**, **d** dan **2.0** menggunakan jenis-jenis pembolehubah yang sesuai. Gabungkan kesemuanya dan paparkan output. Sebagai contoh, **0** telah diisytiharkan menggunakan **byte**, dan **w** telah diisytiharkan menggunakan **char**.

   *Modify the following code in Q1.java so that it displays the output "H3110 w0r1d 2.0". Declare the different variables separately needed to store each of the followings: **H**, **3**, **1**, **0**, **w**, **r**, **d** and **2.0**, using the suitable variable types. Concatenate them and display the output. As an example, **0** is declared using **byte**, and **w** is declared using **char**.*

```
public class Q1 {
   public static void main(String[] args)
   {
      byte zero = 0;
      char alphabetW = 'W';
      String output = ""+alphabetW + zero + alphabetW; //concatenating
      System.out.println(output);
   }
}
```

   Contoh output:

   Sample output:

```
H3110 w0r1d 2.0
```

   (8 markah/*marks*)

2. Tulis satu aturcara untuk permainan dadu antara dua orang pemain. Pertama sekali, aturcara meminta nama pemain-pemain. Seterusnya, aturcara akan meminta pemain pertama untuk melontar dadu. Lontaran pertama menentukan jumlah lontaran yang seorang pemain akan dapat. Aturcara tersebut akan meminta untuk memain melontar berdasarkan jumlah lontaran yang diperolehi daripada skor lontaran pertama. Aturcara akan seterusnya memaparkan jumlah skor (termasuk skor yang didapati daripada lontaran pertama) untuk pemain pertama. Ini akan diulang untuk pemain kedua. Akhirnya, aturcara akan memaparkan pemenang permainan. Pemenang adalah pemain dengan jumlah skor tertinggi.

Write a program for a two players dice game. First, the program asks for the name of the players. Next, the program asks the first player to throw the dice. The first throw determines the amount of throws that a player will get. The program then ask the player to throw based on the score obtained in the first throw. The program will then display the total score (inclusive of the score obtained in the first throw) for the first player. This is repeated for the second player. Finally, the program will display the winner of the game. The winner is the player with the highest score accumulated.

Contoh output:

Sample output:

```
Enter the first player's name: Aznul
Enter the second player's name: Ang

Aznul, press enter to roll your dice!
You obtained 1 roll(s)!
You have to roll 0 more time(s)
Player 1, Aznul obtained a score of: 1

Ang, press enter to roll your dice!
You obtained 4 roll(s)!
You have to roll 3 more time(s)
Ang, press enter to roll your dice!
You obtained 3
Ang, press enter to roll your dice!
You obtained 4
Ang, press enter to roll your dice!
You obtained 2
Player 2, Ang obtained a score of: 13

Player 2, Ang wins!
```

(12 markah/*marks*)

3. Tulis satu aturcara yang menghasilkan 10 nombor rawak, di antara 0 hingga 50. Simpan nombor-nombor ini di dalam satu tatasusunan (*array*), dan paparkan kandungan tatasusunan tersebut. Kemudian, gunakan *bubblesort* untuk mengatur tatasusunan ini dalam turutan menaik. Seterusnya, paparkan kandungan tatasusunan tersebut. Selepas itu, minta satu input nombor daripada pengguna, dan cari nombor tersebut di dalam tatasusunan yang telah diatur tersebut. Kemudian, paparkan indeks input nombor tersebut sekiranya nombor itu wujud, dan paparkan bilangan pusingan (*loop)* yang diperlukan untuk mencari input nombor tersebut. Akhir sekali, simpan tatasusunan yang telah diatur tersebut di dalam fail bernama "Q3.txt". Tulis juga input nombor daripada pengguna dan bilangan pusingan yang diperlukan untuk menemui input nombor tersebut di dalam fail itu.

Write a program that generates 10 random integers, within the range from 0 to 50. Store this integers in an array, and display the content of the array. Then, use bubblesort to sort this array in an ascending order. Next, display the content. Subsequently, ask the user for an input integer, and search the sorted array for this input integer. After that, display the index of the input integer if it exists within the sorted array, and display the number of loops it took to find the input integer. Finally, store the sorted array to a file named "Q3.txt". Write also the input integer from the user along with the number of loops it took to find the input to the file.

Contoh output konsol:

Sample console output:

```
The following is the 10 random numbers generated that is stored in an array:
8 41 46 25 34 16 39 31 32 27
The following is the sorted array (ascending):
8 16 25 27 31 32 34 39 41 46
Enter an integer input to be searched within the sorted array: 27
```

Contoh output di file "Q3.txt" sekiranya input nombor dijumpai.

Sample file output in "Q3.txt" if input integer is found.

```
The content of the sorted array is (ascending): 8 16 25 27 31 32 34 39 41 46
Input integer 27 was found on index 3 of the sorted array.
It was found in 4 number of loops
```

Contoh output di file "Q3.txt" sekiranya input nombor tidak dijumpai.

Sample file output in "Q3.txt" if input integer is not found.

```
The content of the sorted array is (ascending): 8 16 25 27 31 32 34 39 41 46
Input integer 51 was not found in the sorted array.
```

(12 markah/*marks*)

4. Reka bentuk satu kelas ***Account*** yang mengandungi:

- Satu medan peribadi **int** bernama **id** untuk akaun tersebut (tetapan awal adalah 0).
- Satu medan peribadi ***double*** bernama ***balance*** untuk akaun tersebut (tetapan awal adalah 0).
- Satu medan peribadi ***double*** bernama ***annualInterestRate*** yang menyimpan nilai faedah semasa (tetapan awal adalah 0).
- Satu pembina tanpa argumen yang menghasilkan sebuah akaun.
- Satu pembina yang menghasilkan sebuah akaun dengan id dan juga baki akaun (***'balance'***) yang spesifik.
- Kaedah pencapai dan *mutator* untuk id, *balance* dan *annualInterestRate*.
- Satu kaedah bernama ***getMonthlyInterestRate*** yang memulangkan kaedah faedah bulanan.
    - Bahagikan ***annualInterestRate*** dengan 12.
- Satu kaedah bernama ***withdraw*** untuk mengeluarkan wang dengan jumlah yang spesifik, dan memberitahu berapa baki yang ada di dalam akaun selepas pengeluaran.
- Satu kaedah bernama ***deposit*** untuk memasukkan wang dengan jumlah yang spesifik, dan memberitahu berapa baki yang ada di dalam akaun selepas deposit dilakukan.
- Satu kaedah bernama ***displayAccountInfo*** yang memaparkan maklumat sesebuah akaun.

*Design a class named **Account** that contains:*

- *A private **int** data field named **id** for the account (default is 0).*
- *A private **double** data field named **balance** for the account (default is 0).*
- *A private **double** data field named **annualInterestRate** that stores the current interest rate (default is 0).*
- *A **no-argument** constructor that creates an account.*
- *A constructor that creates an account with the **specified** id and initial balance.*
- *The accessor and mutator methods for id, balance, and annualInterestRate.*
- *A method named **getMonthlyInterestRate** that returns the monthly interest rate.*
    - *Divide the **annualInterestRate** by 12*
- *A method named **withdraw** that withdraws a specified amount from the account, and notifies the available balance after the withdrawal is made.*
- *A method named **deposit** that deposits a specified amount to the account, and notifies the available balance after the deposit is made.*
- *A method named **displayAccountInfo** that displays the information of an account*

Gunakan **Q4.java** untuk memeriksa kelas **Account** yang telah anda hasilkan.

*Use **Q4.java** to test the class **Account** that you have created.*

(10 markah/*marks*)

Modifikasi Q4.java untuk menginstan sebuah lagi akaun (cth. acc_b), di mana anda perlu

- Memberi nilai kepada ***annualInterestRate*** sebanyak 8.0.
- Mengset nilai ***balance*** kepada RM20,000.
- Mengset **id** akaun kepada 1001.
- Memaparkan maklumat akaun tersebut.
- Memaparkan kaedah faedah bulanan.
- Melakukan deposit sebanyak RM1,500.
- Melakukan pengeluaran sebanyak RM500.

Modify Q4.java to instantiate another account (e.g. acc_b), where you need to

- *Initialize the* ***annualInterestRate*** *to 8.0.*
- *Set the* ***balance*** *to RM20,000.*
- *Set the* ***id*** *of the account to 1001.*
- *Display the account info.*
- *Display the monthly interest rate.*
- *Perform a deposit of RM1,500.*
- *Perform a withdrawal of RM500.*

Contoh output:

Sample output:

```
The information about account A is as follows:
Account id is: 1234
Account balance is: RM10000.0
Annual interest rate is: 7.77

The information about account B is as follows:
Account id is: 1235
Account balance is: RM20000.0
Annual interest rate is: 8.0
The monthly interest rate is: 0.6666666666666666
Previous balance: RM20000.0
Current balance after deposit of RM1500.0 is: RM21500.0
Previous balance: RM21500.0
Current balance after withdrawal of RM 500.0 is: RM21000.0
```

(8 markah/*marks*)

TAMAT
END