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 2013/2014 : SEMESTER II
ACADEMIC SESSION 2013/2014 : SEMESTER II

WXES1116 : Pengaturcaraan I
Programming I

Jun 2014 Masa: 2 jam
June 2014 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 5 halaman yang dicetak)
(This question paper consists of 4 questions on 5 printed pages)

1. a) Takrifkan satu kelas bernama *Pizza* yang mengandungi medan dan kaedah seperti disenaraikan dalam rajah UML berikut:

Define a class named Pizza that contains the fields and methods as listed in the following UML diagram:

Pizza	
-size : String	This holds the size of the pizza whether large, medium or small
-numOfCheeseTopping : int	This holds the number of cheese toppings
-numOfBeefTopping : int	This holds the number of beef toppings
-numOfChickenTopping : int	This holds the number of chicken toppings
+Pizza ()	
+Pizza (size:String, numOfCheeseTopping : int numOfBeefTopping : int numOfChickenTopping : int)	The constructor with the data fields
+getSize() : String +getNumOfCheeseTopping() : int +getNumOfBeefTopping () : int +getNumOfChickenTopping () : int	Accessors to get the four values
+setSize() : String +setNumOfCheeseTopping () : int +setNumOfBeefTopping () : int +setNumOfChickenTopping () : int	Mutators to set the four values.
+computeCost():double	The *computeCost* method calculates the cost of the pizza : Small : $10 + $2 per topping Medium : $12 + $2 per topping Large : $14 + $2 per topping
+toString() : String	A description of the pizza object to be displayed consisting of the pizza size, quantity of each topping and cost.

(8 markah/*marks*)

b) Tulis satu program, *TestPizza* yang membina satu objek *Pizza* dengan data yang bersesuaian. Paparkan objek tersebut. Satu contoh output diberikan di bawah untuk satu *pizza* besar dengan satu *topping* keju, daging dan ayam.

Write a program, TestPizza that creates a Pizza object with appropriate data. Display the object. A sample output is given below for a large pizza with one cheese, beef and chicken topping.

```
Pizza size       :  large
cheese topping   :  1
beef topping     :  1
chicken topping  :  1

Pizza cost       :  $20.00
```

(3 markah/*marks*)

2. a) Takrif satu kelas bernama *LinearEquation* untuk satu sistem persamaan *linear* 2 x 2:

Define a class named LinearEquation for a 2 x 2 system of linear equations:

$$ax + by = e \quad \text{where} \quad x = \frac{ed-bf}{ad-bc} \;;\; y = \frac{af-ec}{ad-bc}$$
$$cx + dy = f$$

Kelas tersebut mengandungi:

The class contains:

- private data fields a, b, c, d, e, and f.
- a constructor with the arguments for a, b, c, d, e, and f.
- six get methods for a, b, c, d, e, and f.
- a method named **isSolvable()** that returns true if ad – bc is not 0.
- methods named **computeX()** and **computeY()** that return the solution for the equation.

(6 markah/*marks*)

b) Tulis satu program ujian yang meminta pengguna memasukkan nilai untuk a, b, c, d, e, dan f; dan paparkan jawapan. Jika ad - bc adalah 0, paparkan 'persamaan tersebut tiada penyelesaian'.

Write a test program that prompts the user to enter values for a, b c, d, e, and f; and displays the answer. If ad – bc is 0, display 'the equation has no solution'.

(4 markah/*marks*)

3. a) Takrifkan satu kelas bernama *Ship* yang mengandungi:

Define a class named Shipwhich contains:

- a String data field named *shipName*.
- a String data field named *yearBuilt*.
- a no-argument constructor with appropriate default values.
- a constructor that creates *Ship* with the specified data values.
- accessor and mutator methods for the data fields.
- a method **toString()** that returns a string description for the *Ship*.

(4 markah/*marks*)

b) Takrifkan kelas *CruiseShip* yang lanjutkan kelas *Ship*. Ia mengandungi:

Define the CruiseShip class that extends the Ship class. It contains:

- one int data field, *numPassenger*, that holds the maximum number of passengers.
- a no-argument constructor with appropriate default values.
- a constructor that creates *CruiseShip* with the specified data values.
- accessor and mutator methods for the data field.
- a method **toString()** that returns only the ship's name and the maximum number of passengers.

(4 markah/*marks*)

c) Takrifkan kelas *CargoShip* yang lanjutkan kelas *Ship*. Ia mengandungi:

Define the CargoShip class that extends the Ship class. It contains:

- one int data field *cargoCapacity*, that holds the capacity of the cargo.
- a no-argument constructor with appropriate default values.
- a constructor that creates *CargoShip* with the specified data value.
- accessor and mutator methods for the data field.
- a method **toString()** that returns only the ship's name and cargo capacity.

(4 markah/*marks*)

d) Lukis gambarajah UML untuk ketiga-tiga kelas.

Draw UML diagrams for all three classes.

(3 markah/*marks*)

e) Tulis satu program ujian yang membina satu objek *Ship* di mana data diberikan oleh pengguna. Paparkan keputusannya. Begitu juga, bina satu objek *CruiseShip* dan *CargoShip* dengan data yang dibekal pengguna untuk objek tersebut dan paparkan keputusannya.

Write a test program that creates one Ship object with its data provided by the user. Display the results. Similarly, create one CruiseShip object and CargoShip object with user provided data and display the results.

(4 markah/*marks*)

4. a) Takrifkan satu kelas bernama *Position* untuk menyimpan nilai terbesar dan kedudukannya dalam satu tatasusunan 1 dimensi. Kelas ini mengandungi:

Define a class named Position for storing the biggest value and its position in a one-dimensional array. The class contains:

- two public data fields named **whichIndex** and **maxValue** where **whichIndex** stores the index of the biggest element in the array and **maxValue** stores the biggest element in the array.
- a no-argument constructor with default values as 0 and 0.
- a constructor that creates a *Position* with the specified list of positive numbers.
- a method **findMax()** that returns an **instance** of Position:
  public static Position**findMax**(int[] num)

(7 markah/*marks*)

b) Tulis satu program ujian yang membina satu objek *Position* dengan tatasusunan {95,100,74,66} dan paparkan nombor terbesar dalam tatasusunan itu dan kedudukan nombor terbesar tersebut seperti berikut:

Write a test program that creates a Position object with array {95,100,74,66} and displays the biggest number in that array and the position of the number as follows:

```
The biggest element in the array is 100 and its position is at index 1.
```

(3 markah/*marks*)

TAMAT
END