SOALAN 1/*QUESTION 1* (12 MARKAH/*MARKS*)

Diberikan kod Java berikut, tunjukkan dan jelaskan output bagi setiap kod selepas dilaksanakan.
Given the following Java code, show and explain the output of each of the code after executed.

a)
```
public class Vehicle {
   protected int speed;
   private int capacity;

    protected void add(int speed){
       this.speed = speed + 1 ;
    }

    private void add(int speed, int capacity) {
       this.speed = speed + 2;
    }

    protected int speed() {
       return 0;
    }
}
public class Car extends Vehicle {

    public int speed() {
        return this.speed;
    }
}
public class TestingCar {

   public static void main(String[] args)
{
       Vehicle car = new Car();
       int a = 0;
       car.add(120);
      System.out.println(car.speed());
   }
}
```

(4 markah/*marks*)

SOALAN 1/QUESTION 1 (12 MARKAH/MARKS)

Explanation of the code

- **Vehicle Class:**
    - **Attributes:**
        - **speed**: protected int
        - **capacity**: private int
    - **Methods:**
        - **add(int speed)**: Adds the given speed to the current speed and increments the speed by 1.
        - **add(int speed, int capacity)**: Adds the given speed to the current speed and increments the speed by 2.
        - **speed()**: Returns 0.
- **Car Class:**
    - **Extends the Vehicle Class:**
    - **Methods:**
        - **speed()**: Returns the current value of **speed**.
- **TestingCar Class:**
    - **Main Method:**
        - Creates a **Vehicle** object called **car** of the class **Car**.
        - Initializes an integer variable **a** to 0.
        - Calls the **add()** method of the **car** object with a speed of 120. This will call the **add(int speed)** method of the **Vehicle** class.
        - Prints the result of calling the **speed()** method of the **car** object.

Output:

- The **add(120)** method will increase the speed by 1, so the value of speed will be 1.
- The **speed()** method of the **Car** class will return this value.
- The output will be: **1**.

b)
```
String[] data = {"Larry", "Moe", null, "Curly"};
    int sum = 0;

    try {
      for(String s : data)
        sum += s.length();
    }
    catch (Exception exc) { }
    System.out.println(sum);
  }
```

(4 markah/*marks*)

c)
```
public class Overload {

    protected void add(int arg){
       System.out.println("One Parameter : " + arg);
    }

    protected int add(int arg1, int arg2){
       System.out.println("Two Parameters : " + arg1 + " " + arg2);
       return arg1 * arg2;
    }
}

public class DemoOverload {

   public static void main(String[] args) {
       Overload obj1 = new Overload();
       Overload obj2 = new Overload();

       int x = 5;
       int y = 7;

       obj1.add(x+2);
       obj2.add(y+3);
   }
}
```

(4 markah/*marks*)

b) String Array and Sum Calculation

- The code initializes a String array called **data** with 4 elements: **"Larry"**, **"Moe"**, **null**, and **"Curly"**.
- An integer variable **sum** is declared and initialized to 0.
- A **try-catch** block is used to handle potential exceptions during the iteration of the array elements.
- Inside the **try** block, the code iterates through each element **s** of the array **data**.
- For each element, the code adds the length of the string **s** to the **sum** variable.
- If an exception occurs during the loop, the **catch** block will handle it. In this case, it does nothing but catch the exception.
- Finally, the **sum** is printed to the console.

c) Class Overload and Method Overloading

- The code defines a class named **Overload**.
- This class contains two methods with the same name (**add**), but different parameter lists. This is called **method overloading**.
- The first method (**add(int arg)**) takes one integer parameter and prints the message "One Parameter : " followed by the value of the parameter.
- The second method (**add(int arg1, int arg2)**) takes two integer parameters and prints the message "Two Parameters : " followed by the values of the parameters. It then returns the product of the two parameters.
- The code also defines a class named **DemoOverload**.
- This class has a **main** method, which is the entry point of the program.
- Inside the **main** method, two **Overload** objects (**obj1** and **obj2**) are created.
- Two integer variables (**x** and **y**) are declared and initialized.
- Then, **obj1.add(x+2)** and **obj2.add(y+3)** calls the respective overloaded **add** methods of the **Overload** objects. This will print the respective messages and perform the calculation.

Overall

The provided code demonstrates how to use method overloading in Java. The class **Overload** showcases two methods with the same name, allowing for different operations based on the number of arguments passed. The **DemoOverload** class provides a simple example of how to call these overloaded methods.

SOALAN 2/*QUESTION 2* (17 MARKAH/*MARKS*)

Andaikan kelas `A` mempunyai metod `m1()`, `m2()`, dan `m3()`, dan subkelas `B` mempunyai metod `m4()` dan `m5()`.

Suppose a class `A` has methods `m1()`, `m2()`, and `m3()` and a subclass `B` has methods `m4()` and `m5()`.

a) Manakah antara lima kaedah itu boleh diakses oleh objek kelas `A`?
Which of the five methods are accessible by objects of class `A`? (2 markah/*marks*)

b) Manakah antara lima kaedah itu boleh diakses oleh objek kelas `B`?
Which of the five methods are accessible by objects of class `B`? (2 markah/*marks*)

c) Manakah antara lima kaedah itu boleh dipanggil menggunakan pembolehubah bagi jenis `A` yang merujuk kepada objek bagi jenis `B`?
Which of the five methods can be called using a variable of type `A` that refers to an object of type `B`? (2 markah/*marks*)

d) Andaikan `m1()` diisytiharkan sebagai *private.* Manakah antara 5 kaedah itu boleh diakses oleh objek bagi kelas `B`?
Suppose `m1()` is declared private. Which of the five methods are now accessible by objects of class `B`? (2 markah/*marks*)

e) Katakan subkelas `B` yang melanjutkan kelas `A`, melaksanakan *interface* `D` yang melanjutkan *interface* `C`. Antara pernyataan berikut yang manakah adalah **SAH**?
*Suppose that subclass `B` extends class `A` and implements an interface `D` that extends an interface `C`. Which of the following statements are **LEGAL**?*

i. `A obj = new A();`
ii. `B obj = new A();`
iii. `C obj = new A();`
iv. `D obj = new A();`
v. `Object obj = new A();` (5 markah/*marks*)

a) Which of the five methods are accessible by objects of class A?

- m1()
- m2()
- m3()

b) Which of the five methods are accessible by objects of class B?

- m1()
- m2()
- m3()
- m4()
- m5()

c) Which of the five methods can be called using a variable of type A that refers to an object of type B?

- m1()
- m2()
- m3()

d) Suppose m1() is declared private. Which of the five methods are now accessible by objects of class B?

- m2()
- m3()
- m4()
- m5()

e) Suppose that subclass B extends class A and implements an interface D that extends an interface c. Which of the following statements are LEGAL?

- **i. A obj = new A();**
- **ii. B obj = new A();**
- **iii. C obj = new A();**
- **iv. D obj = new A();**
- **v. Object obj = new A();**

f) Terangkan perbezaan utama di antara *inheritance* dan *polymorphism*. Tunjukkan contoh yang sesuai dengan menggunakan kelas `A` dan subkelas `B`.

Describe the main difference between inheritance and polymorphism. Give appropriate examples using class `A` *and subclass* `B`.

(4 markah/*marks*)

SOALAN 3/*QUESTION 3* (16 MARKAH/*MARKS*)

Program berikut tidak dapat dikompil dengan jayanya disebabkan terdapat beberapa ralat. Kenalpasti ralat tersebut dalam baris kod dan kelas berkenaan dan cadangkan penyelesaian untuk membetulkan ralat dalam program ini. Jelaskan jawapan anda.

The following programs cannot be compiled successfully due to some errors. Identify the errors in the code line and related class and suggest a possible solution to fix the errors in the program. Explain your answer.

a)

```
1 class Bank{
2    private int roi;
3
4    abstract public int getROI();
5
6    public void setROI (int roi){
7           this.roi = roi;
8    }
9  }
```

```
1 class CIMB implements Bank{
2
3  public int getROI(){
4    return 5;}
5  }
```

```
1 public class BankROI{
2
3   public static void main (String [] args){
4      Bank b1 = new Bank();
5      b.getROI();
6   }
7 }
```

(6 markah/*marks*)

a) class Bank {

- **Line 4:** *abstract public int getROI()*{ The method getROI is declared abstract and should not have a body.
- **Line 7:** *this.rol = rol;*{ The variable rol is not defined in the class Bank.
- **Line 11:** *public class CIM implements Bank*{ The class CIM should have a constructor that initializes the rol variable.
- **Line 17:** *Bank b1 = new Bank();*{ It's impossible to create an object of type Bank because it is abstract.
- **Line 18:** *b1.getROI();*{ The method getROI is not implemented in the class CIM.

Solution:

- Add the variable rol in the class Bank.
- Implement the abstract method getROI in the class CIM.
- Create a constructor in the class CIM to initialize the rol variable.
- Create an object of type CIM instead of Bank.

```
1  class Bank{
2      private int rol;
3      abstract public int getROI();
4
5      public Bank(int rol) {
6          this.rol = rol;
7      }
8  }
9
10 class CIM implements Bank{
11     private int rol;
12     public CIM(int rol) {
13         this.rol = rol;
14     }
15
16     public int getROI() {
17         return rol;
18     }
19 }
20
21 public class BankMain{
22     public static void main(String[] args) {
23         CIM b1 = new CIM(10);
24         System.out.println(b1.getROI());
25     }
26 }
```

b)

```
1  class Divide{
2
3    public static calculateTheArray (int[] arrayIn, int division) {
4
5    int length = arrayIn.length;
6    int[] arrayOut = new [length];
7
8    for (int i=0; i< length; i++)
9       arrayOut[i] = arrayIn/division;
10
11   return arrayOut;
12  }
13 }
```

```
1  public class DivideArrayProgram {
2
3    public static void main(String[] arg){
4
5   int[] y= {4, 6, 8, 10};
6   int[] z;
7
8   z= calculateTheArray(y,2);
9
10  for (int i=0; i< 3 ; i++)
11      System.out.println(z[i]);
12  }
13 }
```

(10 markah/*marks*)

Code Analysis

The code implements two classes:

- **Divide:** This class has a static method called **calculateTheArray**. The method takes an integer array **arrayIn** and an integer **division** as input. It then creates a new array called **arrayOut** with the same length as **arrayIn**. For each element in **arrayIn**, the method divides the element by **division** and stores the result in the corresponding position of **arrayOut**. Finally, the method returns the **arrayOut**.
- **DivideArrayProgram:** This class contains the main method which demonstrates the usage of the **calculateTheArray** method.
    - It first creates an array **y** with values {4, 6, 8, 10}.
    - It then calls the **calculateTheArray** method with **y** as the input array and 2 as the division value. The result is stored in a new array **z**.
    - Finally, it iterates through the **z** array and prints each element to the console.

Explanation

The code implements a simple array manipulation task. It divides each element of an input array by a given value and stores the result in a new array. This can be useful in various scenarios, such as scaling array values or performing element-wise division in mathematical computations.

Example Usage

Consider the array **y** with values {4, 6, 8, 10}. When calling **calculateTheArray(y, 2)**, the code would divide each element of **y** by 2:

- 4 / 2 = 2
- 6 / 2 = 3
- 8 / 2 = 4
- 10 / 2 = 5

The result is stored in the array **z** with values {2, 3, 4, 5}, which is then printed to the console.

SOALAN 4/*QUESTION 4* (20 MARKAH/*MARKS*)

a) Reka bentuk satu kelas `Staff` yang memegang dan memaparkan maklumat staf termasuk nama, ID dan tarikh diupah. Hasilkan kelas dengan lengkap yang terdiri daripada:

- Tiga pembolehubah: nama staf (`String`), staf ID (`int`) dan tarikh diupah (`String`).
- Satu pembina yang menerima semua nilai nama staf, ID dan tarikh diupah sebagai *arguments* dan menentukan mereka kepada pembolehubah yang sesuai.
- Satu kaedah bernama `displayInfo()` untuk memaparkan semua maklumat staf.

Design a `Staff` class which holds and displays details of staff name, ID and the hire date. Produce the complete class which consists of:

- *Three instance variables: the staff name (a `String`), the staff ID (an `int`) and the hire date (a `String`).*
- *A constructor that accepts all the values of staff name, ID and hire date as arguments and assigns them to the appropriate variables.*
- *A method named `displayInfo()` to display all information of the staff.*

(5 markah/*marks*)

b) Reka bentuk satu kelas `SupportStaff` yang melanjutkan kelas `Staff`. Kelas `SupportStaff` sepatutnya mempunyai pembolehubah untuk memegang gred perkhidmatan seorang staff (`int`) dan jumlah jam kerja lebih masa (`double`).

- Gred perkhidmatan boleh dibahagikan kepada dua: F17 dan N14. Pembolehubah gred perkhidmatan memegang nilai integer yang mewakili gred perkhidmatan staf di mana 1 mewakili perkhidmatan gred F17 dan 2 mewakili perkhidmatan gred N14. Untuk staf gred F17, kadar gaji sejam adalah RM5.00 manakala bagi staff gred N14, kadar gaji setiap jam ialah RM3.50.
- Hasilkan kod bagi satu pembina yang menerima semua nilai gred perkhidmatan dan jumlah jam kerja lebih masa sebagai *argument* dan menentukan mereka kepada pembolehubah yang sesuai.
- Hasilkan kod bagi satu kaedah `calculateOvertimeAllowance()` yang akan mengembalikan hasil pernyataan berikut:-

  jumlah jam kerja lebih masa x kadar gaji setiap jam

Question 4a

Design a Staff Class

- The class has 3 instance variables:
    - **staffName** (String)
    - **staffID** (int)
    - **hireDate** (String)
- **Constructor**:
    - Accepts staffName, staffID, and hireDate as arguments.
    - Assigns the values to the corresponding instance variables.
- **displayInfo() method**:
    - Displays the information of the staff (staffName, staffID, hireDate) in a formatted way.

Question 4b

Design a SupportStaff Class

- **Extends the Staff class.**
- **Instance Variables:**
    - **serviceGrade** (int)
    - **overtimeHours** (double)
- **Constructor**:
    - Accepts serviceGrade, overtimeHours, and all Staff class attributes (staffName, staffID, hireDate) as arguments.
    - Calls the Staff class constructor to initialize the staffName, staffID, and hireDate.
    - Assigns the values of serviceGrade and overtimeHours to the respective instance variables.
- **calculateOvertimeAllowance() method:**
    - Calculates the overtime allowance based on the following formula:
        - overtimeHours * hourlyRate
    - **hourlyRate**:

    - RM5.00 for serviceGrade 1 (F17)
    - RM3.50 for serviceGrade 2 (N14)
- Returns the calculated overtime allowance.

- Hasilkan kod bagi satu kaedah `displayInfo()` untuk memaparkan semua maklumat seorang staf bersama dengan elaun kerja lebih masa yang diterima.

Design a `SupportStaff` *class that extends the* `Staff` *class. The* `SupportStaff` *class should have fields to hold the service grade of a staff (an* `int`*) and hours of working overtime (a* `double`*).*

- *The service grade is divided into two: F17 and N14. The service grade field would be an integer value representing the service grade of the staff where 1 is representing service grade F17 and 2 representing service grade N14. For staffs of grade F17 the hourly pay rate is RM5.00 and for staffs of grade N14 the hourly pay rate is RM3.50.*
- *Produce the code for a constructor that accepts all the values of service grade and hours of working overtime as arguments and assign them to the appropriate fields.*
- *Produce the code for a method named* `calculateOvertimeAllowance()` *which returns the result of the following statement:-*

```
hours of working overtime x hourly pay rate
```

- *Produce the code for a method named* `displayInfo()` *to display the information of the employee along with the overtime allowance received.*

(11 markah/*marks*)

c) Tunjukkan kelas-kelas di atas dengan mewujudkan dua objek dari kelas `SupportStaff` yang memegang data seperti dalam **Jadual 1**:-
Show the classes by creating two objects of `SupportStaff` *class that holds the data as in* ***Table 1****: -*

Jadual/ *Table* 1

Name	ID	Hire Date	Service Grade	Hours of Working Overtime
Clark Kent	123456	1 November 2015	N14	5
Fattah Amin	987654	15 January 2019	F17	8

(4 markah/*marks*)

Requirements:

1. Create a SupportStaff class that extends the Staff class

- The class should have two additional fields:
    - **serviceGrade** (int): to store the service grade of the staff.
    - **overtimeHours** (double): to store the number of overtime hours worked.

2. Implement the serviceGrade field

- The **serviceGrade** should be an integer value representing the service grade of the staff.
- 1 represents **F17** and 2 represents **N14**.

3. Implement the overtimeHours field

- The **overtimeHours** field should be a double value representing the number of overtime hours worked.

4. Implement a constructor for the SupportStaff class

- The constructor should take arguments for all the fields:
    - **name** (String): name of the staff
    - **id** (int): id of the staff
    - **hireDate** (Date): hire date of the staff
    - **serviceGrade** (int): service grade of the staff
    - **overtimeHours** (double): overtime hours worked

5. Implement the calculateOvertimeAllowance() method

- This method should calculate the overtime allowance based on the following formula:
    - **overtimeHours * hourlyPayRate**
- The **hourlyPayRate** should be determined based on the **serviceGrade** as follows:
    - If the **serviceGrade** is **F17**, the **hourlyPayRate** is RM5.00.
    - If the **serviceGrade** is **N14**, the **hourlyPayRate** is RM3.50.

6. Implement the displayInfo() method

- This method should display the information of the staff along with the calculated overtime allowance.
- The output should include the following information:
    - Name
    - ID
    - Hire Date
    - Service Grade
    - Overtime Hours
    - Overtime Allowance

```
import java.util.Date;
2
3public class SupportStaff extends Staff {
4   private int serviceGrade;
5   private double overtimeHours;
6
7   // Constructor
8   public SupportStaff(String name, int id, Date hireDate, int serviceGrade, double overtimeHours) { super(name, id, hireDate);
9      this.serviceGrade = serviceGrade;
10      this.overtimeHours = overtimeHours;
11   }
12
13   // Method to calculate overtime allowance
14   public double calculateOvertimeAllowance() {
15      double hourlyPayRate;
16      if (serviceGrade == 1) { // F17
17         hourlyPayRate = 5.00;
18      } else if (serviceGrade == 2) { // N14
```

```
19          hourlyPayRate = 3.50;
20      } else {
21          hourlyPayRate = 0.00; // Default case
22      }
23      return overtimeHours * hourlyPayRate;
24  }
25
26  // Method to display staff information
27  public void displayInfo() {
28      double overtimeAllowance = calculateOvertimeAllowance();
29      System.out.println("Name: " + getName());
30      System.out.println("ID: " + getId());
31      System.out.println("Hire Date: " + getHireDate());
32      System.out.println("Service Grade: " + serviceGrade);
33      System.out.println("Overtime Hours: " + overtimeHours);
34      System.out.println("Overtime Allowance: RM" + overtimeAllowance);
35  }
```

SOALAN 5/ *QUESTION 5* (35 MARKAH/ *MARKS*)

Reka bentuk satu aplikasi `Coffee System` yang akan meminta pengguna untuk memberi input bilangan beg kopi yang dijual, berat unit beg, harga kopi per kilo di dalam ringgit dan cukai jualan seperti yang ditunjukkan dalam **Rajah 1**.

Design an application named `Coffee System` *that will ask users to input number of coffee bags sold, the unit weight of a bag of coffee, the price of coffee per kilograms and the sales tax as shown in* ***Figure 1***.

a) Hasilkan kod bagi kelas `CoffeeBag` yang lengkap. Kelas anda perlu mengandungi:
Produce the code for a complete `CoffeeBag` *class . Your class should have:*

- Semua pembolehubah yang berkenaan / *All related instance variables;*
- Satu konstruktor menerima bilangan beg kopi yang dijual, berat unit beg dan harga kopi per kilogram sebagai parameter / *A constructor that accepts number of coffee bags sold, the unit weight of a bag of coffee and the price of coffee per kilogram as parameters;*
- **Tiga (3)** metod /**THREE (3)** methods
  - `setTaxRate()`: Untuk menetapkan nilai kadar cukai/ *To set the value of the tax rate*
  - `calculateTotalPrice()`: Untuk mengira jumlah harga / *To compute the total price*

    ```
    totalPrice = bagWeight x  numberOfBag x  pricePerKg;
    ```
  - `calculatePriceWithTax()`: Untuk mengira jumlah harga dengan cukai/ To compute the total price with tax

    ```
    totalPriceWithTax = totalPrice + totalPrice x TaxRate;
    ```

(10 markah/*marks*)

b) TuliskanHasilkan kod bagi satu kelas `CoffeeBagGUI` yang akan menggunakan kelas `CoffeeBag` di 4(a) sebagai enjin pemprosesan. `CoffeeBagGUI` ini perlu ada satu butang di mana pengguna boleh klik selepas data dimasukkan dan pengiraan perlu dilaksanakan. Data perlu dimasukkan melalui medan teks, dengan label bermaklumat yang jelas. Jumlah harga dengan cukai juga akan dipaparkan dalam medan teks, tetapi ianya harus ditetapkan sebagai *un-editable*. Contoh antaramuka `CoffeeBagGUI` adalah seperti yang dipaparkan di bawah: