|  |  |
| --- | --- |
| **Assignment Case** |  |
| COMP6153001  Operating System |
| **Computer Science** | **O223-COMP6153-CH01-01** |
| ***Valid on*** *Odd Semester Year 2021/2022* | **Revision 00** |

1. Seluruh mahasiswa tidak diperkenankan untuk:

*All students are not allowed to:*

* + - Berdiskusi dan/atau bekerja sama dengan mahasiswa lainnya

*Discuss and/or work together with other student participants*

* + - Melihat sebagian atau seluruh jawaban mahasiswa lain

*Seeing a part or the whole answer from another student*

* + - Membuka dan menyalin dari **BUKU** atau **CATATAN**, **VIDEO** dari pengajar (recording kelas, VBL, Youtube, dsb) dan **REFERENSI** lainnya

*Open and copy from any resources such as notes, videos (class recording, VBL, Youtube, etc) and other references*

* + - Membuka dan menyalin jawaban dari internet (google, stackoverflow, dsb)

*Open and copy answer from the internet (google, stackoverflow, etc)*

* + - Mengerjakan soal yang tidak sesuai dengan tema yang ada di soal,

*Working with another theme which is not in accordance with the existing theme in the matter of the case,*

* + - Melakukan tindakan kecurangan lainnya,

*Committing other dishonest actions,*

* + - Secara sengaja maupun tidak sengaja melakukan segala tindakan kelalaian yang menyebabkan hasil karyanya berhasil dicontek oleh orang lain / kelompok lain.

*Accidentally or intentionally conduct any failure action that cause the results of the project was copied by someone else / other groups.*

1. Jika mahasiswa terbukti melakukan tindakan seperti yang dijelaskan butir 1 di atas, maka **nilai mahasiswa** yang melakukan kecurangan (menyontek maupun dicontek) akan di – **NOL** – kan.

*If the student is proved to the actions described in point 1 above, the score of the student which committed dishonest acts (cheating or being cheated) will be “Zero”*

1. Perhatikan jadwal pengumpulan jawaban, segala jenis pengumpulan jawaban di luar jadwal tidak dilayani.

*Pay attention to the submission schedule, all kinds of submission outside the schedule will not be accepted*

1. Bila Anda tidak membaca peraturan ini, maka Anda dianggap telah membaca dan menyetujuinya

*If you have missed to read these regulations, so you are considered to have read and agreed on it*

1. Persentase penilaiaan untuk matakuliah ini adalah sebagai berikut:

*Marking percentage for this subject is described as follows:*

|  |  |
| --- | --- |
| **Tugas Mandiri**  *Assignment* | **UAP**  *Final Exam* |
| 40% | 60% |

1. Software yang digunakan pada matakuliah ini adalah sebagai berikut:

*Software will be used in this subject are described as follows:*

|  |
| --- |
| **Software**  *Software* |
| VMware Workstation 15  VM Ubuntu Client 20.04  Java 8  Eclipse 2020.6  NachOS 5.0j |

## Ekstensi file yang harus disertakan dalam pengumpulan tugas mandiri, dan uap untuk matakuliah ini adalah sebagai berikut:

*File extensions should be included in assignment collection for this subject are described as follows:*

|  |  |
| --- | --- |
| **Tugas Mandiri**  *Assignment* | **UAP**  *Final Exam* |
| DOCX, JAVA, CLASS | JAVA, CLASS |

## Soal

*Case*

**DietCHeck**

CHae Seo Jin really loves eating. But as a K-drama actress, she needs to be careful with her diet. Otherwise, it will affect her health. Due to that reason, CHae always makes sure to keep track of her food intake. As a programmer, CHae wants you to help her create a **program** that will calculate how CHae's foods affect her health and weight. However **Before programming** the application you will need to **create** a **bash command line** to do these tasks:

1. From the **home directory**, write a **command line** to create a directory based on the hierarchy above in **a single execution**!

<Home Directory>

-- Food

|-- Food Type

| |-- Junk Food

| |-- Healthy Food

|-- Nutritional Facts

| |-- Calories

| |-- Nutritions

|-- Ingredients

1. From the **Nutritional Facts folder**, write a **command line** to create two new files named **“BurnFats.txt”** and **“notFats.txt”** inside **Calories folder**. The **first file (BurnFats.txt)** will contains the **text "Burn your fats!"**. Then, set the **permission** for **both file** with details as :

**Owner :** Read, Write, Execute

**Group :** Read, Write

**Others :** Read, Write

Note that this must be done in **a single execution**.

1. From the **Nutritional Facts folder**, write **a single command line** to show the **name of all files** with the following criteria:

**- Inside Calories folder**

**- The types are .txt**

**- File is not empty**

1. Write a **single command line** to **display all processes** for **all users**, then **sort the process by PID in ascending order**.
2. Java Programming

Now here is where it gets fun, You are tasked by CHae to **program** an application that will calculate how CHae's foods affect her health and weight using **Java Programming Language** with these following concepts:

* + - 1. Abstract Class

You need to design at least **three** classes, **one abstract** class, and **two concrete** classes. Abstract class consists of all **common** attributes and behavior that both of concrete class had. Concrete class consist of **specific** attribute and behavior that not common between the concrete classes

1. Encapsulation

To **hide** the data of a class from an **illegal** direct access, all of the attributes of the class must be **encapsulated** and will be accessed using an **accessor** and **mutator** that may perform validation before accessing the encapsulated attribute

1. Inheritance

All of the concrete class **must inherit all** attribute and behavior from the abstract class

1. Polymorphism

Both concrete Classes **must override** a specific **inherited** behaviour (method) that is inherited from the **abstract** class.

The following are the specifications for the program:

* + - At the start of the application, the program will prompt user to **input user’s weight**. Validate that the weight must be **between 38 and 50 (inclusive)**.

DietCHeck!

============

Input your weight in kg [38 - 50] >> 32

Input your weight in kg [38 - 50] >> 40

Figure 1. Input Weight

* + - Then, the program will **calculate user’s health, minimum weight, and maximum weight** with following specification:

**Health = 100**

**Minimum Weight = 80% \* Weight**

**Maximum Weight = 120% \* Weight**

Figure 2. Health & Weight Formula

* + - After all the required input has been filled, the program will show the **full detail** of the user.

Current Data

============

Weight : 40.0 kg

Health : 100.0%

Be careful with your weight.

Minimum weight 32.0 kg, maximum weight 48.0 kg.

Press [Enter] to proceed...

Figure 3. Show User Data

* + - After that, The program will **display main menu** that consist of 3 menus which are:
* **Insert Food Intake**
* **View Intake History**
* **Exit**

Diet CHeck!

============

1. Insert Food Intake

2. View Intake History

3. Exit

>>

Figure 4. Main Menu

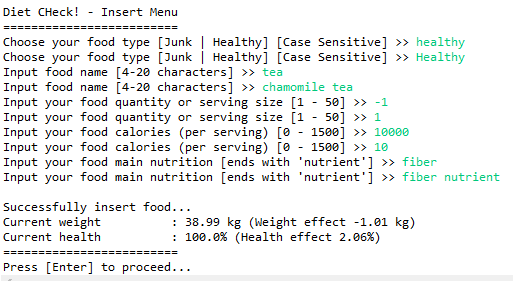
(Don’t forget to validate that **user input** must be **between 1 to 3 inclusively**).

* + - If the user choose **menu 1 (“Insert Food Intake”)** , this program will require the user to input information about the data of the food that follows the following details which are:
* **Food Type**. Validate that food type must be either **“Junk”** or **“Healthy” (case sensitive)**.
  + - **Food Name**. Validate that the length must be **between 4 to 20 characters**.
    - F**ood quantity or serving size**. Validate that food quantity must be **between 1 to 50 (inclusive)**.
    - F**ood calories (per serving)**. Validate that the food quantity must be between **0 to 1500 (inclusive)**.
    - If food type is **“Healthy”**, prompt user to **input food main nutrition**. Validate that the food main nutrition **must end with** **“nutrient”**.
    - If food type is **“Junk”**, prompt user to **input food sugar level**. Validate that the sugar level must be **between 0 to 5 (inclusive)**.

**Figure 10. Input Sugar Level**

* + - If food type is **“Junk”**, check if **health is greater than 85** and **current weight is more than 4 kg lighter** from user’s starting weight. If the condition is met, offer user to have a cheat day. Validate that the input must be either **‘y’ or ‘n’ (case sensitive)**.

**Figure 11. Input Cheat Day**



*Figure 5. Healthy Food*

You maintain your health & weight well. You deserve a cheat day.

Do you need a cheat day? [y | n] [case sensitive] >> yes

Do you need a cheat day? [y | n] [case sensitive] >> y

*Figure 6. Cheat Day*

* User will **gain weight** and **gain health** from Healthy Food. If Healthy Food contains **fiber** as its nutrition, the food will cause user to **lose weight**. Calculate the **Healthy Food effects** with the following formula:

**Weight Effect = (Calories \* Quantity \* 0.1%) + 1**

**Health Effect = (Calories \* Quantity \* 0.6%) + 2**

Figure 7. Healthy Food Effects Formula

* User will **gain weight** and **lose health** from Junk Food. Calculate the **Junk Food effects** with the following formula:

**Weight Effect = Calories \* (Sugar Level + 1) \* Quantity \* 0.2%**

**Health Effect = Calories \* (Sugar Level + 1) \* Quantity \* 0.5%**

Figure 8. Junk Food Effects Formula

* If user is allowed for a **cheat day**, **decrease food calories by 20% and sugar level to 50%**. Note that the calories and sugar level will be decreased first before the Junk Food affects user’s health and weight.

**New Calories = 80% \* Old Calories**

**New Sugar Level = 50% \* Old Sugar Level**

Processing cheat day. Calories reduced to 513.6, sugar level to 0.5

Figure 9. Cheat Day Formula

* **Update user’s weight and health** based on the calculation. Validate that **health cannot go lower than 0 and higher than 100**. Tell the user that the food has been successfully inserted and **show user’s current weight and health**.

Successfully insert food...

Current weight : 38.41679999999999 kg (Weight effect 2.0 kg)

Current health : 91.148% (Health effect -5.0%)

=========================

Press [Enter] to proceed...

Figure 10. Success Insert Food

* There will be some conditions where the **diet fail**. Make sure that user’s **health is not lower than 40 and current weight is between minimum and maximum weight**. If such conditions happen, stop the program, and tell the user that the diet fails. Show user’s **final weight and health**.

Oops! Your diet fails ):

Thank you for using Diet CHeck.

===============================

Your final weight : 50.27 (10.27 kg from your starting weight)

Your final health : 74.3% (25.7% decreased)

Figure 11. Diet Fail

* If the user choose **menu 2 (“View Intake History”)** **show all the food** that user had insert previously, categorized from the food type. **Show the food name and its weight effect**. If there are no data, tell user that it is **empty**.

Diet CHeck - Healthy Food History

=================================

0. Empty

Diet CHeck - Junk Food History

=================================

1. Nasi Padang | 4.0 kg weight effect

2. Indomieeee | 1.191 kg weight effect

Press [Enter] to proceed...

Figure 12. View Intake History

* If user chooses the third menu option, the program will stop. **Show user’s final weight and health, also show the difference between starting and final data.**

Thank you for using Diet CHeck.

===============================

Your final weight : 45.8 (5.8 kg from your starting weight)

Your final health : 85% (15% decreased)

Figure 13. Exit Menu