FACULTY OF COMPUTING
UTM Johor Bahru

SECJ1023 Programming Technique II

Semester 2 2023/2024

Group Project Deliverable 4:

Project Finale

Lecturer: Ms. Lizawati binti Mi Yusuf

Group Member:

Name:	Matric Number:
Ch’ng Seng Hong	A23CS0058
Foo Ming Kuang	A23CS5026
Tan Qing Qing	A23CS5034

Table of Content

Section A

Project Description 3

Analysis and Design

Flow Chart 5

UML Digram 6

Section B:

OO Concept in system:

Encapsulation 7

Composition 10

Aggregation 11

Inheritance 12

Polymorphism 13

Array of objects 14

Exception handling 15

Reference 17

Section A

Project Description:

In today's fast-paced world, having a good sleep always going ignore by many of the people nowadays due to demanding schedules and routines. Yet, sleep is a fundamental of health, it is essential for physical recovery, mental well-being and overall quality of life. Recognizing the importance of sleep, the Dream Catcher system has been developed as a sleep cycle analyser that designed to monitor and enhance sleep quality. Dream Catcher offers a comprehensive solution to analyse and improve sleep patterns, ensuring users achieve the restful sleep they need.

The primary purpose of Dream Catcher is to enhance the well-being and health of its users by providing a thorough understanding of their sleep patterns. By leveraging detailed data and personalized insights, Dream Catcher aims to help users achieve better sleep quality, thereby improving their overall health and daily performance.

Dream Catcher is designed to provide a detailed analysis of sleep duration and quality throughout the night. By monitoring the user's sleep cycle, the system offers a comprehensive overview of sleep health. This detailed analysis allows users to gain insights into how long they spend and the quality of their sleep, which is crucial for making informed decisions about their sleep habits.

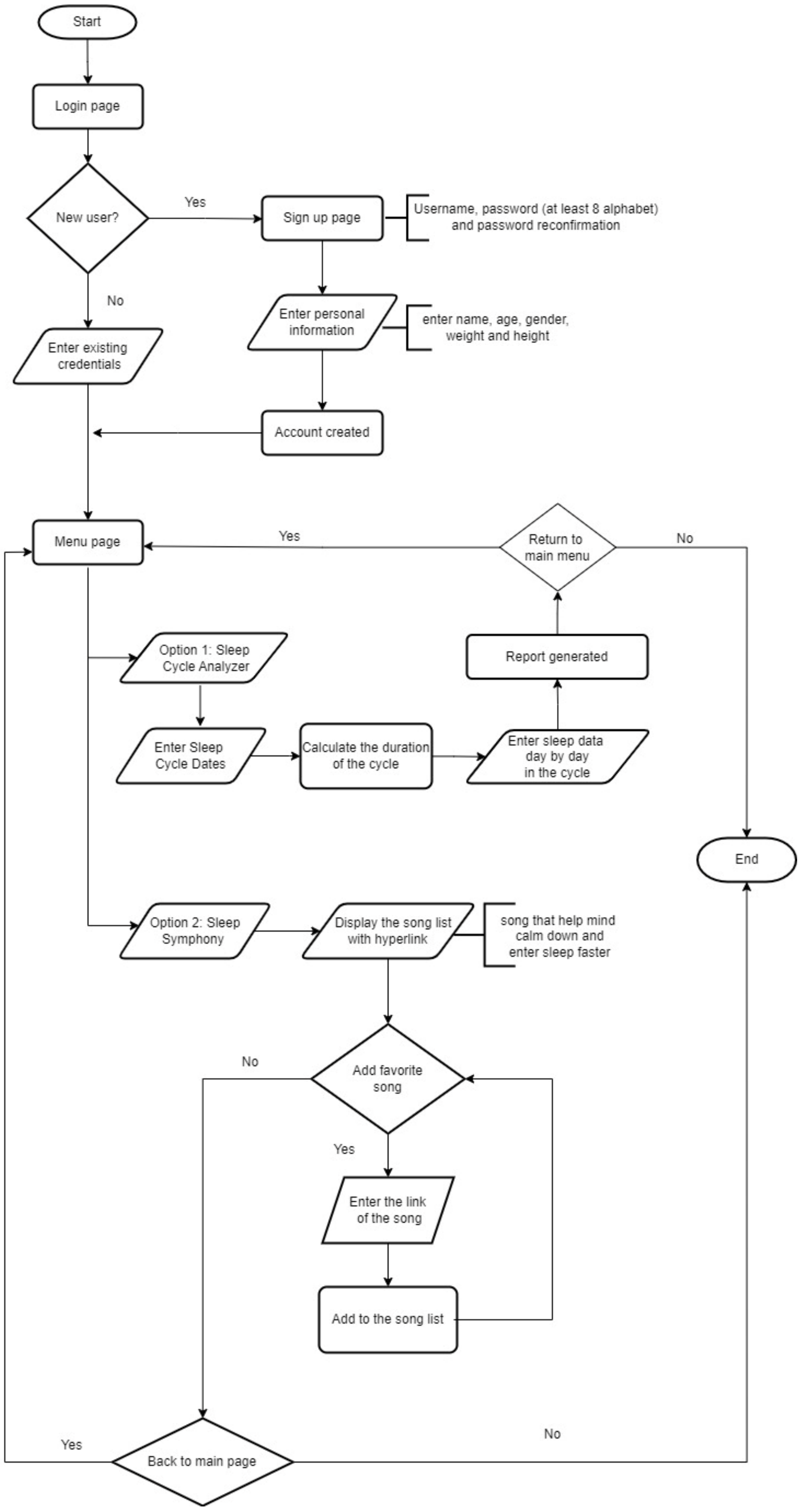
Poor sleep is associated with a myriad of health problems, including obesity, heart disease, and mental health disorders. Dream Catcher aims to reduce the risk of these health issues by encouraging healthy sleep habits. By promoting better sleep, the system contributes to overall health and well-being, enhancing not only physical health but also cognitive function and emotional stability.

A standout feature of Dream Catcher is the Sleep Symphony function, designed to enhance the sleep experience through the power of music. Sleep Symphony offers two categories of soothing sounds: classical music and white noise. Both categories are scientifically proven to promote relaxation and improve sleep quality. Classical music's calming melodies can help slow down the mind and body, making it easier to drift into a deep sleep. White noise, on the other hand, can mask disruptive sounds and create a consistent auditory backdrop conducive to restful sleep.

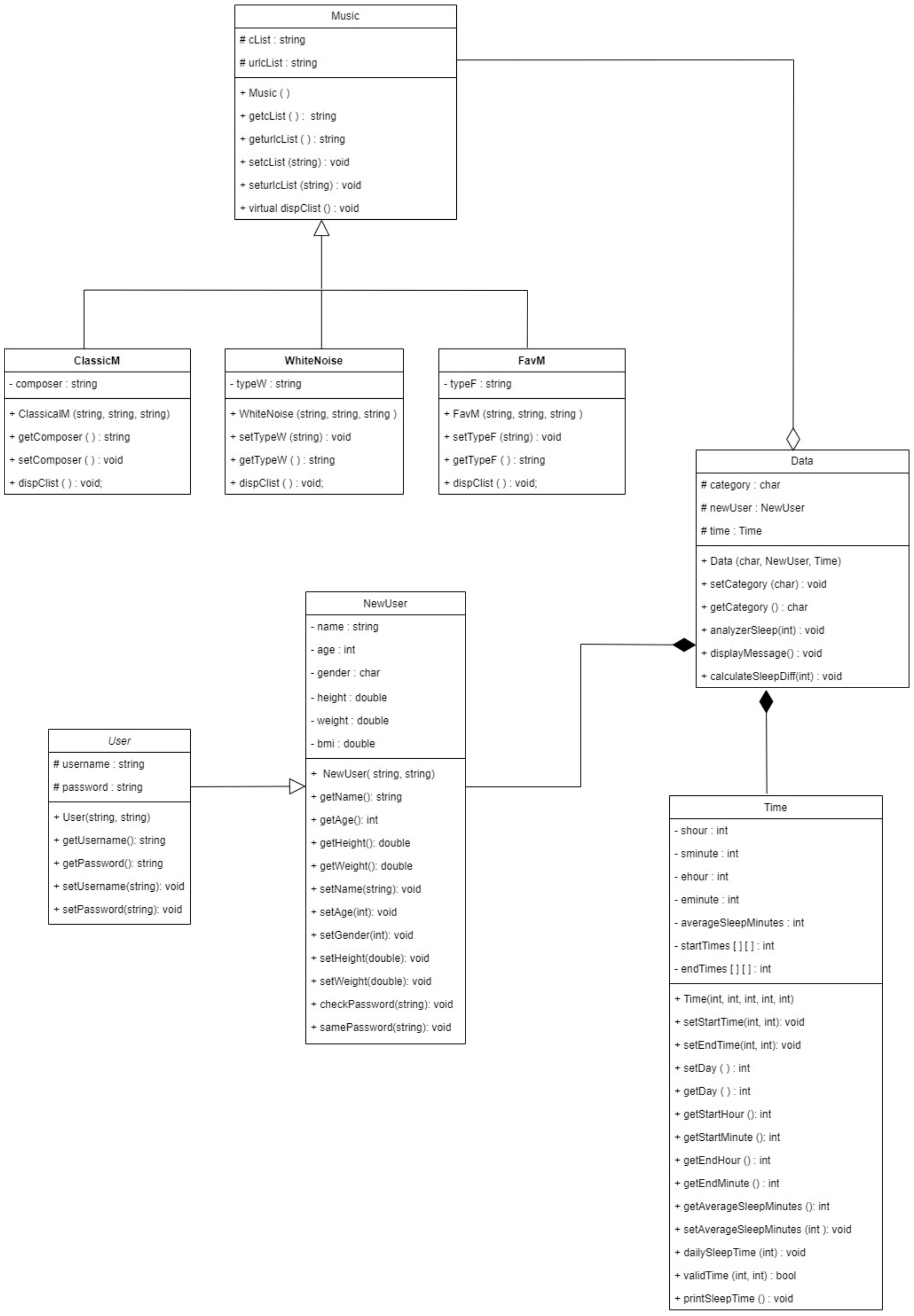
Moreover, Dream Catcher allows users to personalize their sleep soundtrack by adding their favourite music. Users can simply enter the name of the music and its hyperlink, and Dream Catcher will incorporate it into their Sleep Symphony playlist. This personalization ensures that users have a customized and enjoyable sleep experience tailored to their preferences.

By providing analysis, Dream Catcher empowers users to take control of their sleep and, consequently, their overall health. The innovative Sleep Symphony function further enhances the sleep experience, making Dream Catcher not just a sleep analyser but a comprehensive sleep improvement system. With Dream Catcher, users can look forward to waking up refreshed, rejuvenated, and ready to tackle the day ahead.

Flow chart:


Start
Login page
New user?
Yes
Sign up page
Username, password (at least 8 alphabet) and password reconfirmation
Enter personal information
enter name, age, gender, weight and height
No
Enter existing credentials
Account created
Menu page
Yes
Return to main menu
No
Option 1: Sleep Cycle Analyzer
Report generated
Enter Sleep Cycle Dates
Calculate the duration of the cycle
Enter sleep data day by day in the cycle
End
Option 2: Sleep Symphony
Display the song list with hyperlink
song that help mind calm down and enter sleep faster
No
Add favorite song
Yes
Enter the link of the song
Add to the song list
Yes
Back to main page
No

UML Diagram:


Music
# cList : string
# urlcList : string
+ Music ( )
+ getcList ( ) : string
+ geturlcList ( ) : string
+ setcList (string) : void
+ seturlcList (string) : void
+ virtual dispClist () : void
ClassicM
- composer : string
+ ClassicalM (string, string, string)
+ getComposer ( ) : string
+ setComposer ( ) : void
+ dispClist ( ) : void;
WhiteNoise
- typeW : string
+ WhiteNoise (string, string, string)
+ setTypeW (string) : void
+ getTypeW ( ) : string
+ dispClist ( ) : void;
FavM
- typeF : string
+ FavM (string, string, string)
+ setTypeF (string) : void
+ getTypeF ( ) : string
+ dispClist ( ) : void;
Data
# category : char
# newUser : NewUser
# time : Time
+ Data (char, NewUser, Time)
+ setCategory (char) : void
+ getCategory () : char
+ analyzerSleep(int) : void
+ displayMessage() : void
+ calculateSleepDiff(int) : void
NewUser
- name : string
- age : int
- gender : char
- height : double
- weight : double
- bmi : double
+ NewUser(string, string)
+ getName(): string
+ getAge(): int
+ getHeight(): double
+ getWeight(): double
+ setName(string): void
+ setAge(int): void
+ setGender(int): void
+ setHeight(double): void
+ setWeight(double): void
+ checkPassword(string): void
+ samePassword(string): void
User
# username : string
# password : string
+ User(string, string)
+ getUsername(): string
+ getPassword(): string
+ setUsername(string): void
+ setPassword(string): void
Time
- shour : int
- sminute : int
- ehour : int
- eminute : int
- averageSleepMinutes : int
- startTimes [ ] [ ] : int
- endTimes [ ] [ ] : int
+ Time(int, int, int, int, int)
+ setStartTime(int, int): void
+ setEndTime(int, int): void
+ setDay ( ) : int
+ getDay ( ) : int
+ getStartHour (): int
+ getStartMinute (): int
+ getEndHour () : int
+ getEndMinute () : int
+ getAverageSleepMinutes (): int
+ setAverageSleepMinutes (int): void
+ dailySleepTime (int) : void
+ validTime (int, int) : bool
+ printSleepTime () : void

Section B

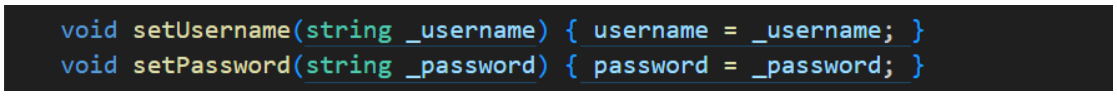
Object-Oriented Concept involved in:

Encapsulation:

We involved the encapsulation concept in the system. Encapsulation restricts direct access to an object's internal state by exposing only what is necessary through public methods. This prevents unauthorized access and modification of data. In the DreamCatcher system we have 7 class which is:

- ***User class***

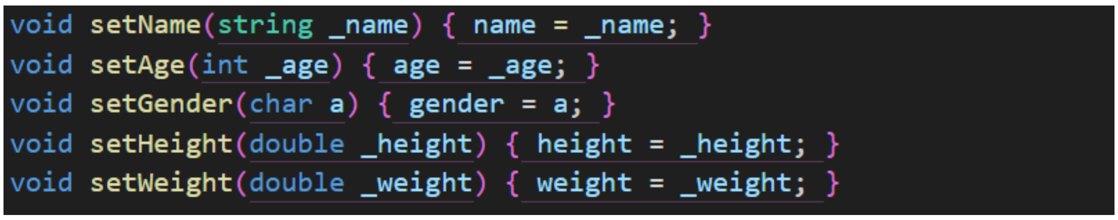
  In User class, there two protected data which is username and password. This class use to handle the username and password that enter by the user. When the user sign in, the program will compare the username and the password that enter by the user with the username and password in the users.txt



```
    void setUsername(string _username) { username = _username; }
    void setPassword(string _password) { password = _password; }
```

- ***NewUser class***

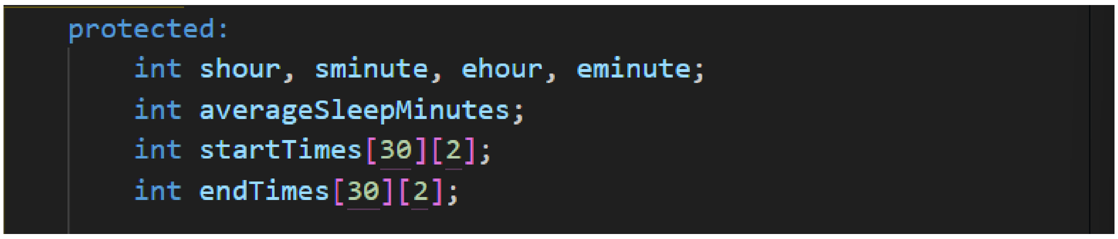
  Inside the NewUser class, we get input from the user such as name, gender, weight and height, store the private data by using mutator functions. By obtaining this value, the program is able to analyse the sleep and give the rating of the sleep more precisely and accurately.



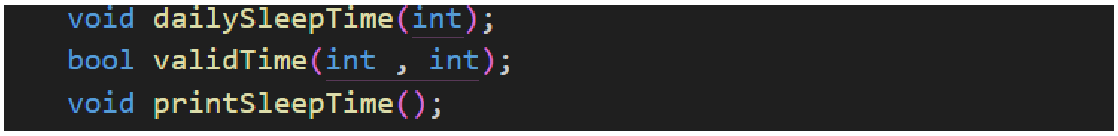
```
void setName(string _name) { name = _name; }
void setAge(int _age) { age = _age; }
void setGender(char a) { gender = a; }
void setHeight(double _height) { height = _height; }
void setWeight(double _weight) { weight = _weight; }
```

- ***Time class***

In this class, we multiple attributes to calculate the duration of time that the sleep cycle of the user. Program will request the user to enter the Start Sleep time and the End Sleep time for the calculation of the sleep time. Also in this class, we have the method to print the results which is the daily sleep time



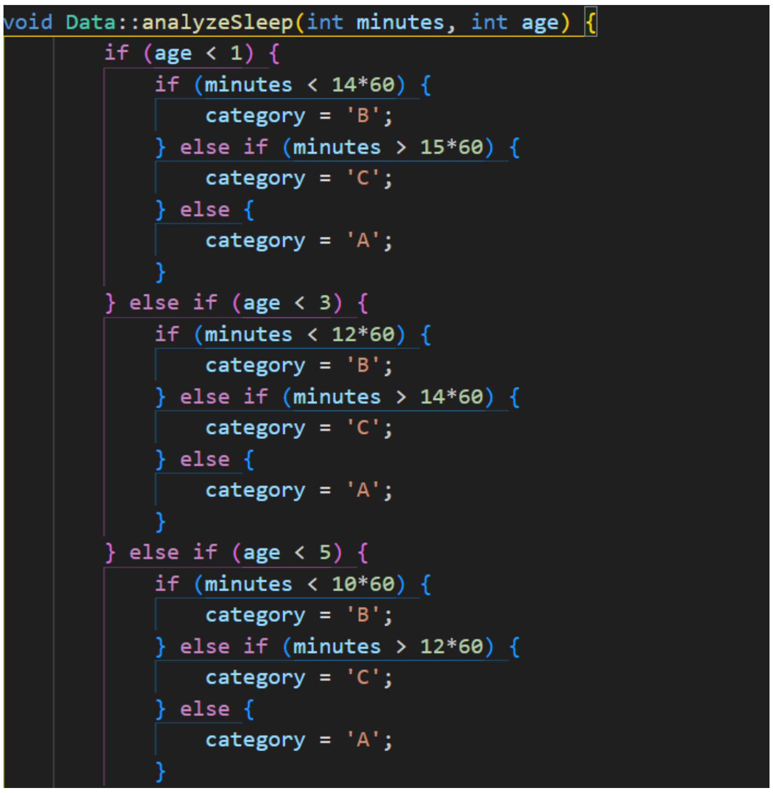
```
protected:
    int shour, sminute, ehour, eminute;
    int averageSleepMinutes;
    int startTimes[30][2];
    int endTimes[30][2];
```



```
void dailySleepTime(int);
bool validTime(int , int);
void printSleepTime();
```

- ***Data class***

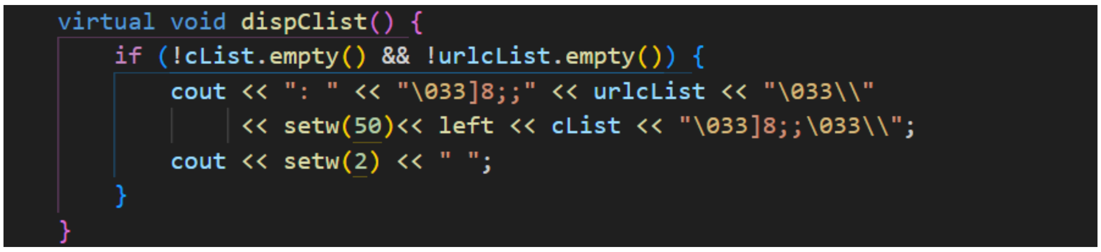
Inside the data class, the category which is the attribute the class is to determine the sleep quality of the user. It categorises the duration of sleep and show the output when the whether the user have good sleep, poor sleep or overslept.



```
void Data::analyzeSleep(int minutes, int age) {
    if (age < 1) {
        if (minutes < 14*60) {
            category = 'B';
        } else if (minutes > 15*60) {
            category = 'C';
        } else {
            category = 'A';
        }
    } else if (age < 3) {
        if (minutes < 12*60) {
            category = 'B';
        } else if (minutes > 14*60) {
            category = 'C';
        } else {
            category = 'A';
        }
    } else if (age < 5) {
        if (minutes < 10*60) {
            category = 'B';
        } else if (minutes > 12*60) {
            category = 'C';
        } else {
            category = 'A';
        }
```

- ***Music class***

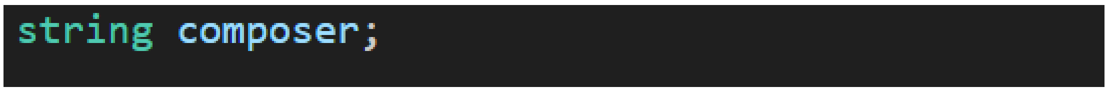
Inside the Music class, we have two attributes to store name of the music and the link of the music that input form the musiclist.txt. To simplify the output, the link and the name has been combined as one element. The program will check whether the link and the name of music is empty or not before output the data to prevent compilation.



```
virtual void dispClist() {
    if (!cList.empty() && !urlcList.empty()) {
        cout << ": " << "\033]8;;" << urlcList << "\033\\"
            << setw(50)<< left << cList << "\033]8;;\033\\";
        cout << setw(2) << " ";
    }
}
```

- ***ClassicM class***

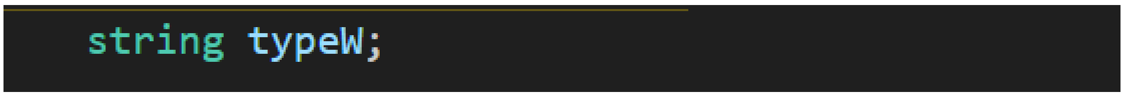
ClassicM class is used to store the classical song which can help slow down the mind and body, making it easier to drift into a deep sleep. Compared to Music class, it has an extra attribute that used to store the composer's name of the music



```
string composer;
```

- ***WhiteNoise class***

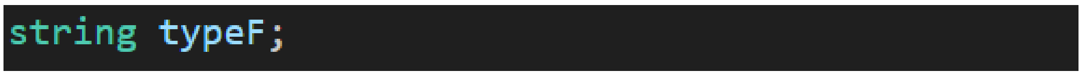
WhiteNoise class used to store the white noise which can mask disruptive sounds and create a consistent auditory backdrop conducive to restful sleep. Same as classic class, there is an extra attribute but it is used to store the type of white noise.



```
string typeW;
```

- ***FavM class***

FavM class is used to store the music that added by the user. In FavM, there are an extra attribute compared to the Music class, which is the type of music.



```
string typeF;
```

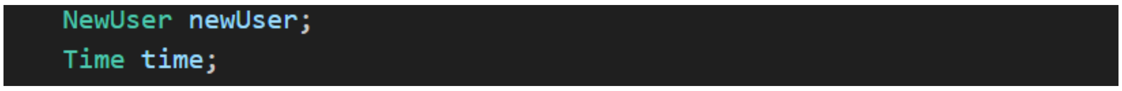
Composition:

- ### Data class and Time class

User object has a Time object because when we create the existence of user object and time object are independent. One of them destroyed will not affect the other one. We set this relationship is because the user can work independently without the time class and vice versa. This Time object doesn't affect the user object, we just use the Time object to calculate the duration of the cycle by using the input data (start date and end date of cycle). Thus, it doesn't require to destroy Time object when the User object destroy.

- ### NewUser class and Data class

NewUser object consists of Data object, both are dependent. When Data class object is destroyed. The NewUser object will also be destroyed. The data object cannot work independently without the presence of use object. We established this relationship to protect the data of the new user, and the data class object requires attributes from the new user to generate the sleep cycle report.

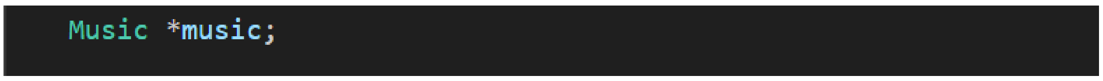


```
    NewUser newUser;
    Time time;
```

Aggregation:

- **NewUser class and Music class**

  NewUser object has a Music object. When the newuser object is destroyed, the Music object doesn’t affect. This is because the Music object provides a list of music tracks that help the user fall asleep faster and doesn’t require any data from the new user. Since the user able to add their favourite music but this is directly added to the music library. That means that when the newuser added music the other newuser will share the same library. Thus, the Music cannot be destroyed when the newuser object destroyed.



```
    Music *music;
```

Inheritance:

- **User class and NewUser class**

  User class is a general class with basic user attributes where NewUser class is a more specific class that inherits from User time, adding additional personal details for example weight and height. Thus, NewUser is a User.

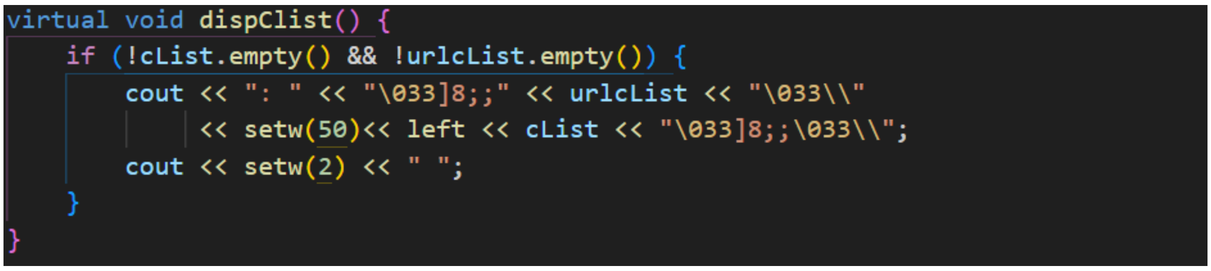
- **Music class, ClassicM class, WhiteNoise class and FavM class**

  ClassicM class, WhiteNoise class and FavM class is a Music class. It enables the user to add their favourite music into the program. Instead of there are only 10 classical music and 10 white noises, user can add their favourite music in the FavM class. Three of them are all inherit because they inherit the attribute name of music and link of music from the Music class to construct object with an extra attribute.

Polymorphism:

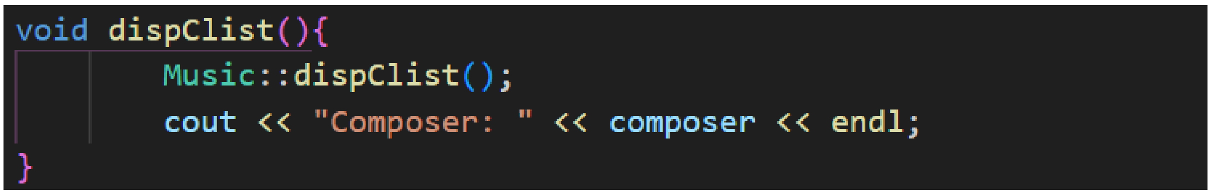
In the program, we include the polymorphism to ensure that the same code can be reused for objects of different types. This reduces code duplication and promotes reuse. We overridden the code of display output of the Music class, ClassicM class, WhiteNoise class and the FavM class to show the music name and link component with an extra attribute in the child's classes.

-In Parent class (Music class)

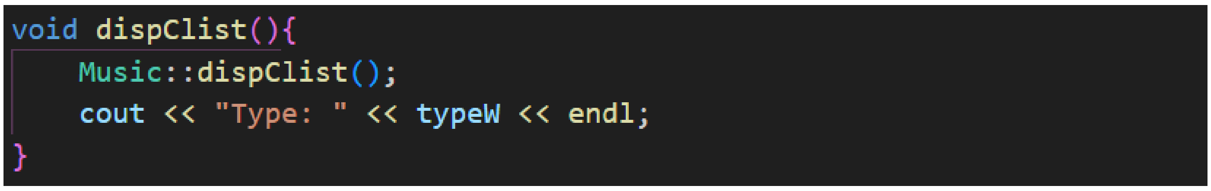


```
virtual void dispClist() {
    if (!cList.empty() && !urlcList.empty()) {
        cout << ": " << "\033]8;;" << urlcList << "\033\\"
            << setw(50)<< left << cList << "\033]8;;\033\\";
        cout << setw(2) << " ";
    }
}
```

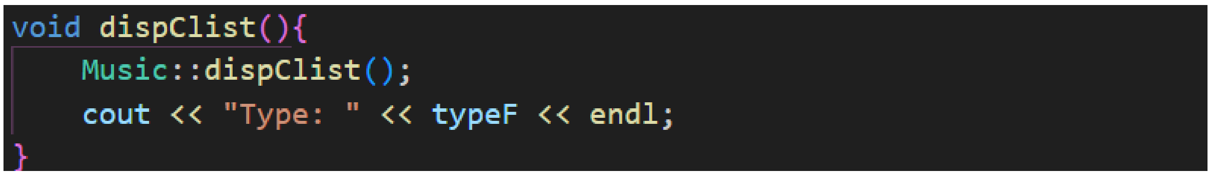
- In child class (ClassicM class, WhiteNoise class and FavM class)



```
void dispClist(){
        Music::dispClist();
        cout << "Composer: " << composer << endl;
}
```



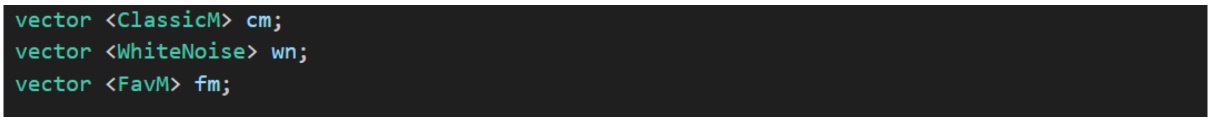
```
void dispClist(){
    Music::dispClist();
    cout << "Type: " << typeW << endl;
}
```



```
void dispClist(){
    Music::dispClist();
    cout << "Type: " << typeF << endl;
}
```

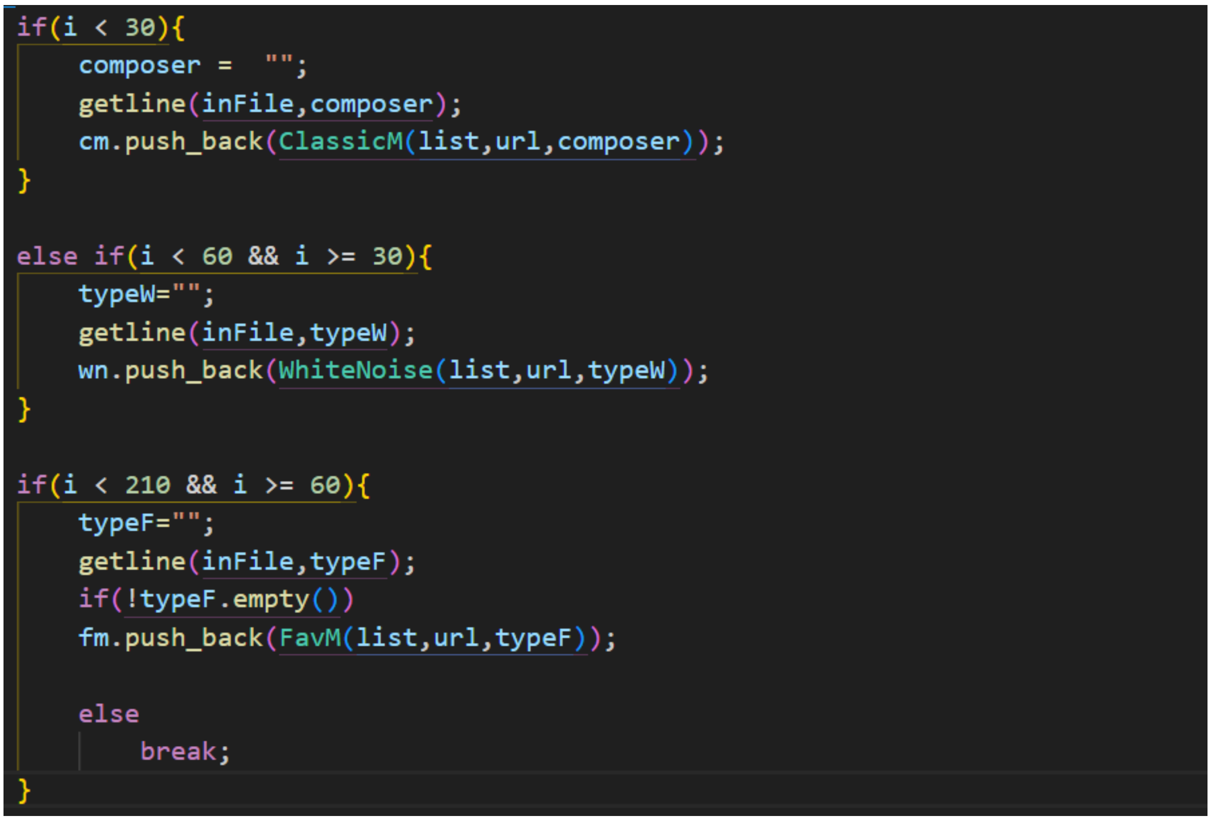
Array of Object:

For the sleep symphony, we involved the array object. We defined vector for every child class of the sleep symphony which is:



```
vector <ClassicM> cm;
vector <WhiteNoise> wn;
vector <FavM> fm;
```

Every song name and link together with the extra attribute that we input file from the musiclist.txt will be used to construct an object and add into vector of the respectively class.



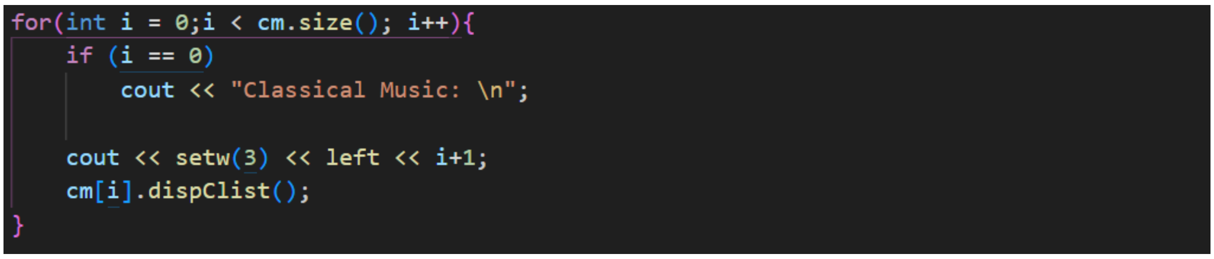
```
if(i < 30){
    composer =  "";
    getline(inFile,composer);
    cm.push_back(ClassicM(list,url,composer));
}

else if(i < 60 && i >= 30){
    typeW="";
    getline(inFile,typeW);
    wn.push_back(WhiteNoise(list,url,typeW));
}

if(i < 210 && i >= 60){
    typeF="";
    getline(inFile,typeF);
    if(!typeF.empty())
    fm.push_back(FavM(list,url,typeF));

    else
        break;
}
```

The vector will show the output by using the array method :

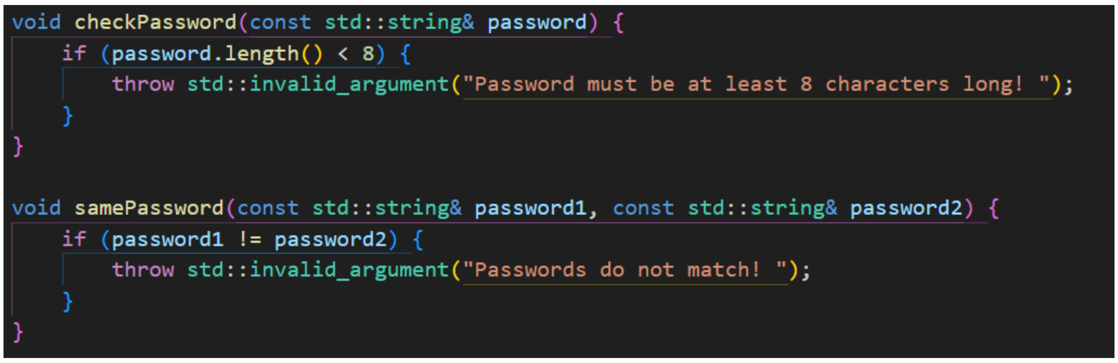


```
for(int i = 0;i < cm.size(); i++){
    if (i == 0)
        cout << "Classical Music: \n";

    cout << setw(3) << left << i+1;
    cm[i].dispClist();
}
```

Exception Handling:

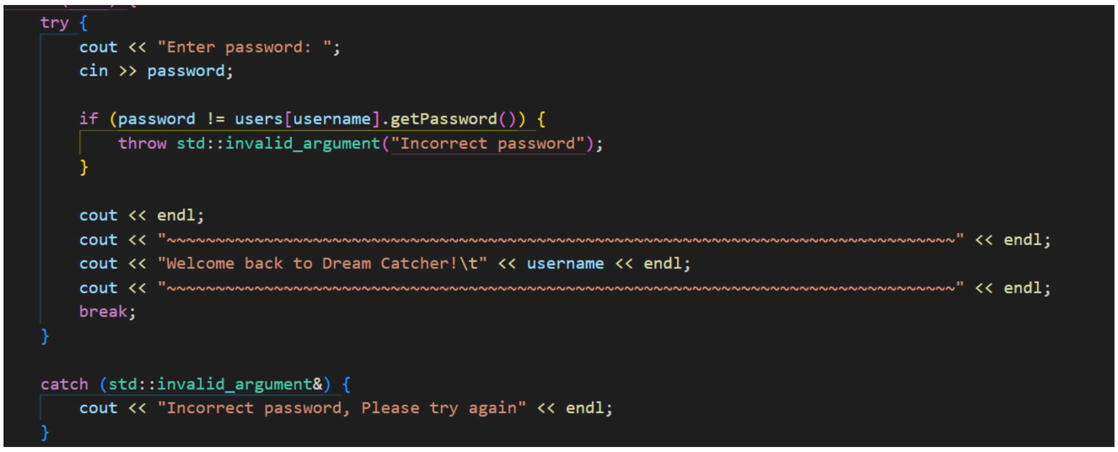
We apply the exception handling for the user’s password to double confirm the password to make sure that the password user first time enter is the correct one. This is to prevent the careless mistake that make by the user. For the password, we still have another exception handling which is to restrict the user to create a password that is at least 8 character. By increasing the length of the password can may increase the security of the account.



```cpp
void checkPassword(const std::string& password) {
    if (password.length() < 8) {
        throw std::invalid_argument("Password must be at least 8 characters long! ");
    }
}

void samePassword(const std::string& password1, const std::string& password2) {
    if (password1 != password2) {
        throw std::invalid_argument("Passwords do not match! ");
    }
}
```

Besides the we also involved exception handling in the sign in page of the program. When the user enter the incorrect password, the user will require to re-enter the password before we redirecting the user to the main page of the program.



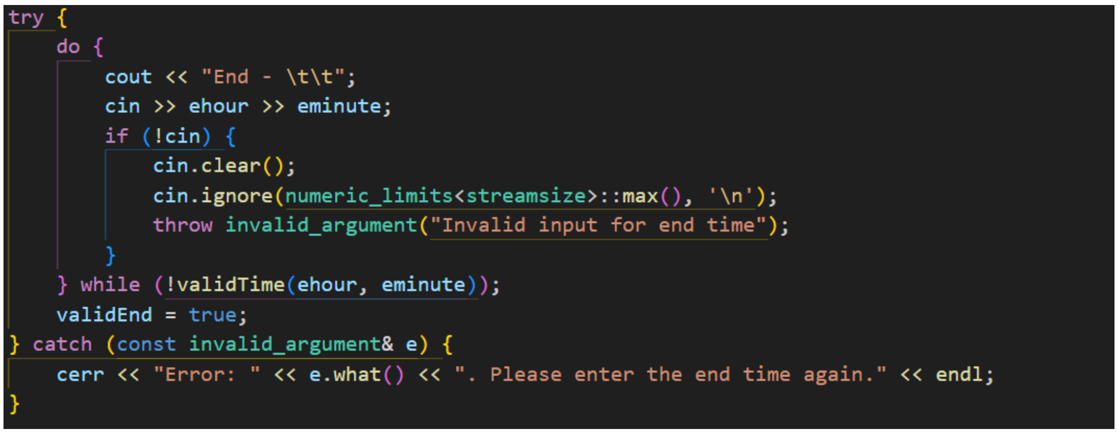
```cpp
    try {
        cout << "Enter password: ";
        cin >> password;

        if (password != users[username].getPassword()) {
            throw std::invalid_argument("Incorrect password");
        }

        cout << endl;
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
        cout << "Welcome back to Dream Catcher!\t" << username << endl;
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
        break;
    }

    catch (std::invalid_argument&) {
        cout << "Incorrect password, Please try again" << endl;
    }
```

For the sleep time, to ensure the data that provide by the user for calculation are in the correct fomat, we also involved the except handling in this part to restrict the user enter maximum integer number is 24 for the hour part and maximum integer is 59 for the minute part.



```
try {
    do {
        cout << "End - \t\t";
        cin >> ehour >> eminute;
        if (!cin) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            throw invalid_argument("Invalid input for end time");
        }
    } while (!validTime(ehour, eminute));
    validEnd = true;
} catch (const invalid_argument& e) {
    cerr << "Error: " << e.what() << ". Please enter the end time again." << endl;
}
```

Reference:

Olson, Eric J. “How Many Hours of Sleep Do You Need?” *Mayo Clinic*, 21 Feb. 2023, www.mayoclinic.org/healthy-lifestyle/adult-health/expert-answers/how-many-hours-of-sleep-are-enough/faq-20057898.