Name: Ai Yoshida
Neptun code: FPYYT9
Class: The Basic of Programming2
Professor: Vaitkus Márton

Period Tracker Documentation for Developer

Data structure

- three original header files (“Period.h”, “PMS.h”, “file.h”)
- using object oriented by using three classes to naminulate this program.
- using file handling for saving PMS data and Period data.
- using exception handling in the main function to let users avoid input different values.
- using dynamic data allocation for objects of class PMS, Period.
- “using namespace std” is a unified namespace in this whole code(including all original header files.)

class, algo, testing, techniques

Header Files

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <fstream>
```

Files

```
"Period.h"// for class Period
"PMS.h"// for class PMS
“file.h”// for class file
```

classes

class PMS

[Purpose of this class]
This class’s attributes are for the values of PMS data.
This class’s methods are for getting attributes of class PMS.

private attributes

```
        string PMS_all;
                // to store all values at the same time so that it is easy to save value in the
        text file.
        int date_year;
                //date
        int date_month;
        int date_day;
        int body_condi;
                // to store the level of body condition from 0 to 9.
```

public methods:

```
    PMS() //constructor
         cin>> PMS_all;
                  //This takes input from the user as string.
        string temp_year = PMS_all.substr(0,4);
                 //This temp_year takes only year part from string
        string temp_month = PMS_all.substr(5, 2);
                 //Takes only month part from PMS_all
        string temp_day = PMS_all.substr(8, 2);
                 // do the same
        string temp_body_condi = PMS_all.substr(11, 1);
                 // do the same

        date_year= stoi(temp_year);
        date_month = stoi(temp_month);
        date_day = stoi(temp_day);
        body_condi = stoi(temp_body_condi);
                 //These stoi functions will convert int from string, so that PMS attribute int
values can have their values.

PMS(string str){
                 // copy constructor.
        PMS_all = str;
                 // store all values as string.
        string temp_year = PMS_all.substr(0,4);
        string temp_month = PMS_all.substr(5, 2);
        string temp_day = PMS_all.substr(8, 2);
        string temp_body_condi = PMS_all.substr(11, 1);
                 //these temporary string values are stored separately.
        date_year= stoi(temp_year);
        date_month = stoi(temp_month);
        date_day = stoi(temp_day);
        body_condi = stoi(temp_body_condi);
                 //these stoi functions convert string value to int value.

PMS(){};
                 // destructor

string str_getter(){return PMS_all;}
int date_year_getter(){return date_year;}
int date_month_getter(){return date_month;}
int date_day_getter(){return date_day;}
int body_condi_getter(){return body_condi;}
                 // these methods get private attributes and return them.
```

class Period

[Purpose of this class]

This class’s attributes are for the values of Period data.
This class’s methods are for getting attributes of class Period.

private attributes:

```
    string Period_start;
            //to store start date value as string (YYYY/MM/DD)
     string Period_end;
            //to store end date value as string (YYYY/MM/DD)
     string str_body_condi;
            //to store body condition level value as a string(0-9)
     string all;
            //to store all data as string (to store value to text file easier)
     int start_date_year;
     int start_date_month;
     int start_date_day;
     int end_date_year;
     int end_date_month;
     int end_date_day;
     int body_condi;
            //those int values are input by constructor from string values. so that it makes
it easier to manipulate these values as int.
```

public methods:

```
Period()
            //construcor
     {
        cout<<"Please enter the start date of period [YYYY/MM/DD]"<<endl;
        cin>> Period_start;
        cout<<"Please enter the end date of period[YYYY/MM/DD]"<<endl;
        cin>> Period_end;
        cout <<"Please enter the body condition [0(Good)-9(Bad)]"<<endl;
        cin>> str_body_condi;

        all += Period_start;
        all += Period_end;
        all += str_body_condi;
            // all is a string value which has all data (for file writing)

        string s_temp_year = Period_start.substr(0,4);
        string s_temp_month = Period_start.substr(5, 2);
        string s_temp_day = Period_start.substr(8, 2);
            //here make values separately to input them to int variables.

        start_date_year= stoi(s_temp_year);
        start_date_month = stoi(s_temp_month);
        start_date_day = stoi(s_temp_day);
            //stoi function convert data type from string to int
            string e_temp_year = Period_end.substr(0,4);
            string e_temp_month = Period_end.substr(5, 2);
            string e_temp_day = Period_end.substr(8, 2);
                    //same thing for end date

            end_date_year= stoi(e_temp_year);
            end_date_month = stoi(e_temp_month);
            end_date_day = stoi(e_temp_day);
            body_condi = stoi(str_body_condi);
                    //same thing
        }

Period(string str)
                    //2nd constructor with a string argument. Doing the same things with 1st
constructor above.
        {
            string body_condi_temp;
            all = str;
            Period_start = str.substr(0,10);
            Period_end = str.substr(10,10);
            body_condi_temp = str.substr(20,1);

            string s_temp_year = Period_start.substr(0,4);
            string s_temp_month = Period_start.substr(5,2);
            string s_temp_day = Period_start.substr(8,2);

            start_date_year= stoi(s_temp_year);
            start_date_month = stoi(s_temp_month);
            start_date_day = stoi(s_temp_day);
            body_condi = stoi(body_condi_temp);

            string e_temp_year = Period_end.substr(0,4);
            string e_temp_month = Period_end.substr(5,2);
            string e_temp_day = Period_end.substr(8,2);

            end_date_year= stoi(e_temp_year);
            end_date_month = stoi(e_temp_month);
            end_date_day = stoi(e_temp_day);
        }

        ~Period(){}
                    // deconstrucor

        string str_getter(){return all;}
        string start_date_getter(){return Period_start;}
        string end_date_getter(){return Period_end;}
        int body_condi_getter(){return body_condi;}
```

//getters for private attributes.

class file

[Purpose of this class]

- This class has only public methods. These methods manipulate file handling of data of class PMS and Period.
- Therefore, this header file also include “PMS.h” and “Period.h”

```
public:
   void add_PMS(string str)
   {
       ofstream ofs("PMS.txt", ios::out|ios::app);
       ofs << str << "\n";
       cout<<"The data is saved successfully!"<< endl;
       ofs.close();
   }
                  //This method takes “string str” as an argument, and write it into “PMS.txt”
                  //ofs…to open and write to the “PMS.txt”
                  //by using ios::out|ios::app, it always opens an existing text file, and adds
         value.


   void read_PMS()
   {
       string str[256];
       string line;
       int j = 0;
       ifstream f("PMS.txt");
       int length;

       while(getline(f, line)) //get one sentence each time until there is no line in the file
       {
           str[j]=line;
           j++;
       }
       f.close();
       length = j+1;
       cout<<"------------------------------------"<<endl;
       cout <<"List of PMS date"<<endl;
       cout<<"------------------------------------"<<endl;
        if (j==0)
        {
           cout<<"The list is empty"<<endl;
       }
       else
```

```
    {
        for(int i=0; i<length;i++)
        {
            PMS *C = new PMS(str[i]);
            cout<<"Date:"<<C->PMS::date_year_getter();
            cout<<"/"<<C->PMS::date_month_getter();
            cout<<"/"<<C->PMS::date_rday_getter()<<endl;
            cout<<"Body condition: "<<C->PMS::body_condi_getter()<<endl;
            delete C;
        }
    }
}
                //This void method reads and displays the Period values list from “Period.txt”.


void add_Period(string str)
{
    ofstream ofs("Period.txt", ios::out|ios::app);
    ofs << str << "\n";
    cout<<"The data is saved successfully!"<< endl;
    ofs.close();
}
                //This method takes “string str” as an argument, and write it into “Period.txt”
                //ofs…to open and write to the “Period.txt”
                //by using ios::out|ios::app, it always opens an existing text file, and adds
      value.
 void read_Period()
{
    string str[256];
    string line2;
    ifstream f("Period.txt");
    int body_condi_temp2 = 0;
    int j = 0;

    while(getline(f, line2))
    {
        str[j]=line2;
        j++;
    }
    f.close();

    cout <<"List of period date"<<endl;
    cout<<"-----------------------------------"<<endl;
    if (j==0){
        cout<<"The list is empty"<<endl;
    }else{
    for(int i=0; i<j+1;i++)
    {
        Period *Cp = new Period(str[i]);
        cout<<"Period date:"<<Cp->Period::start_date_getter();
        cout<<"～"<<Cp->Period::end_date_getter()<<endl;
        cout<<"Body condition: "<<Cp->Period::body_condi_getter()<<endl;
        delete Cp;
    }
    }
    cout<<"------------------------------------"<<endl;
}
                //This void method reads and displays the PMS values list from “PMS.txt”.
```

menu function

[Purpose of this function]
This menu function firstly shows the menu interface to the user. Then the user will input the number from 1 to 4, as it says. int choice will receive the number, and it leads to the if statement depending on what number is input to the choice.

first, if statement, in the case choice ==1

```
        {
          PMS *A = new PMS;
                //create instance dynamically
          file *B = new file;
                //create file instances dynamically.
          B->file::add_PMS(A->PMS::str_getter());
                //this file method will do the adding work
          delete A;
          delete B;
                //by using dynamic allocation here, it can repeat.
          }
```

second if statement, in the case choice ==2

```
        {
          Period *Ap = new Period;
          file *Bp = new file;
          Bp->file::add_Period(Ap->Period::str_getter());
          delete Ap;
          delete Bp;
          }
                //basically, doing the same thing, Period class version.
```

third if statement, in the case choice is 3.

```
        {
          file *B3 =  new file;
          B3->file::read_Period();
```

```
delete B3;
}
```
//it creates a file instance, and it leads the read_Period() method to read file period values and display.

Fourth statement, choice is 4.

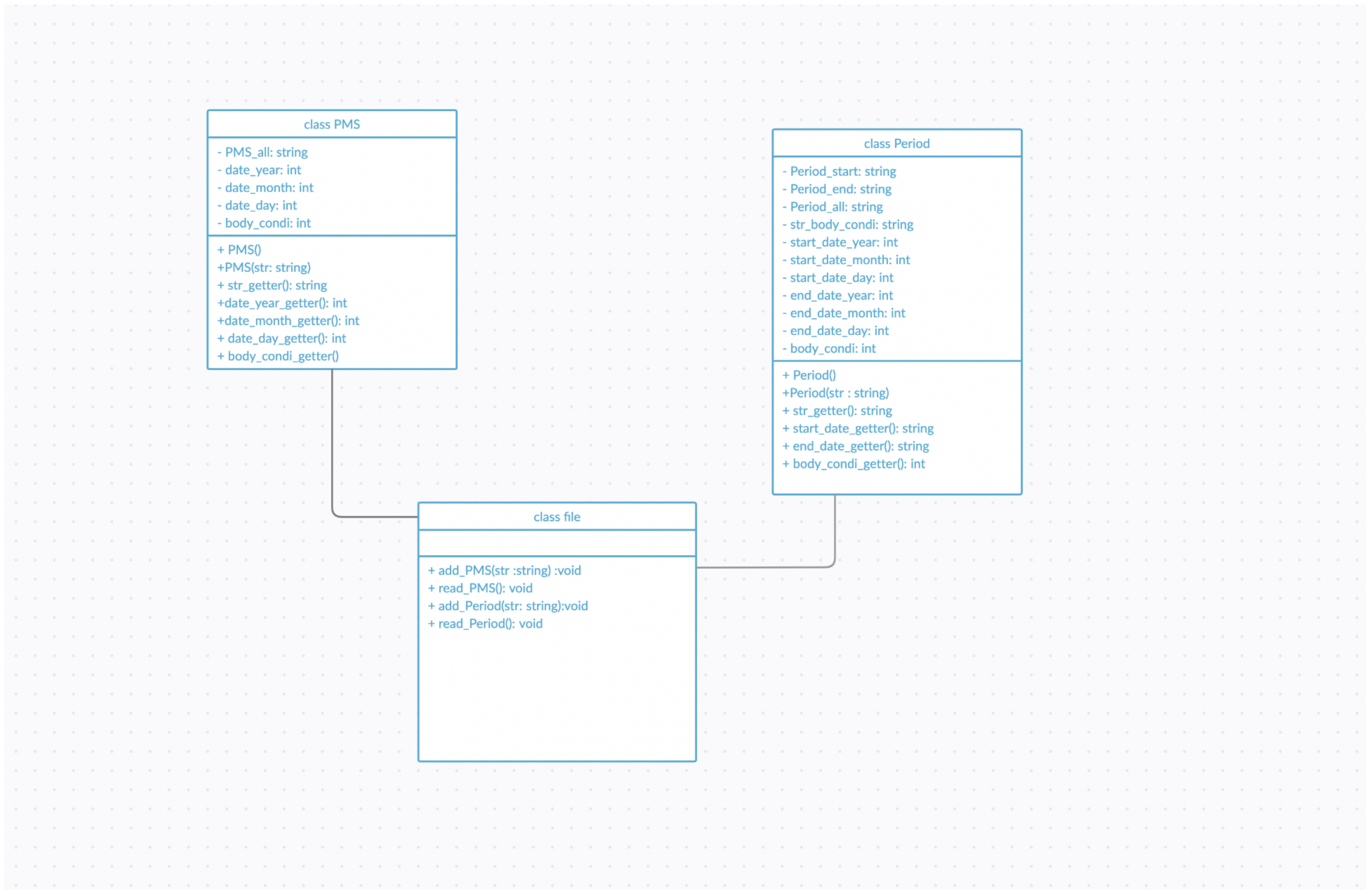
```
{
file *B4 = new file;
B4->file::read_PMS();
delete B4;
}
```
//it creates a file instance, and it leads the read_PMS() method to read file PMS values and display.

algorithms

This program mainly uses if statement and for statement to move the program.
Order is

1. text files are created in the main function, if they do not exist yet.
2. menu function is called, and first menu is displayed to the users, and wait choice input (if statement)
3. depends on the choice, classes are called and dynamically allocated.
4. if one series is done, this program finishes.

UML diagram


class PMS
- PMS_all: string
- date_year: int
- date_month: int
- date_day: int
- body_condi: int
+ PMS()
+PMS(str: string)
+ str_getter(): string
+date_year_getter(): int
+date_month_getter(): int
+ date_day_getter(): int
+ body_condi_getter()
class Period
- Period_start: string
- Period_end: string
- Period_all: string
- str_body_condi: string
- start_date_year: int
- start_date_month: int
- start_date_day: int
- end_date_year: int
- end_date_month: int
- end_date_day: int
- body_condi: int
+ Period()
+Period(str : string)
+ str_getter(): string
+ start_date_getter(): string
+ end_date_getter(): string
+ body_condi_getter(): int
class file
+ add_PMS(str :string) :void
+ read_PMS(): void
+ add_Period(str: string):void
+ read_Period(): void