3 Example : TimeOfDay class

Here we want to develop a class "TimeOfDay", which will be an abstract data type for time of day .

3.1 Class Design

- The Class manages hours and minutes. Hour is in 24-hours notation.
  - private member variables: **hours** and **minutes**

```
// hours
int hours;
// minutes
int minutes;
```

- constructors
  - A constructor takes **hours** and **minutes** as input

```
TimeOfDay(int hours, int minutes);
// precondition : hours and minutes are integers.
// Initializes the time according to the arguments.
// call checktime() to convert a right form.
```

  - A constructor takes **minutes** as input

```
TimeOfDay(int minutes);
// precondition : minutes is integers.
// Initializes the time according to the arguments.
// call checktime() to convert a right form.
```

    * call a private function "**checktime()**" to check time is in a proper form.
  - Default constructor (takes no inputs) set time 00:00

```
TimeOfDay();     // Initialize the time to 00:00
```

- private member function "**checktime()**" checks if **hours** and **minutes** satisfy the criteria:
  - **hours** must be 0,1,2,...22,23 : $0 \leq$ **hours** ≤ 23
  - **minutes** must be 0,2,3...58,59 : $0 \leq$ **minutes** ≤ 59
    * If **minutes** < 0, add 60 to **minutes** and subtract 1 from **hours,** repeat those until **minutes** ≥ 0.
    * If **minutes** > 59, subtract 60 from **minutes** and add 1 to **hours** repeat those until **minutes** ≤ 59.

```
void checktime();
// precondition : hours and minutes are set
// Modify hours to be between 0 and 23.
// Modify minutes to be between 0 and 59.
// * If minutes<0, add 60 to minutes and subtract 1 from hours,
//   repeat those until minutes >=0.
// * If minutes>59, subtract 60 from minutes and add 1 to hours,
//   repeat those until minutes <= 59.
```

- accessors
  - To get **hours**, public member function

```
int get_hours() const;
// return hours
```

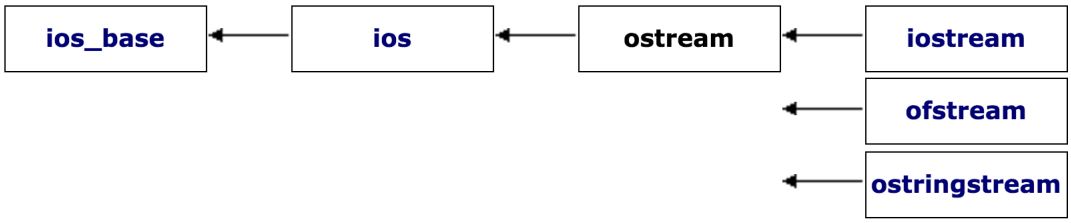
    * Since this function won't change member variables, it is better to have "**const**" modifier.
  - To get **minutes**, public member function

```
int get_minutes() const;
// return minutes
```

    * Since this function won't change member variables, it is better to have "**const**" modifier.
  - To print time to a stream object, public member function

```
void output(std::ostream &strm) const;
// Output time to stream
```

    * Since this function won't change member variables, it is better to have "**const**" modifier.
    * Note that "**ostream**" is the base class of "**iostream**" and "**ofstream**"



    * We can send **std::ofstream** object or **std::cout** to **ostream** object "strm"
      · **std::cout** is a object of "**ostream**" class.

- Friend functions
  - Friend function, "**equal**", which compares two **TimeOfDay** objects and returns **true** if those are same, otherwise returns **false**.

```
friend bool equal(const TimeOfDay& t1, const TimeOfDay& t2);
// precondition : t1 and t2 have values.
// returns true if t1 and t2 represents the same time;
// otherwise,  return false
```

    * Since two **TimeOfDay** objects, we use **friend** function
  - Friend function, "**add**", which adds two **TimeOfDay** objects and returns a new object of **TimeOfDay** that stores the result.

```
friend TimeOfDay add(const TimeOfDay& t1, const TimeOfDay& t2);
// precondition : t1 and t2 have values.
// returns t1 + t2
```

    * Since two **TimeOfDay** objects, we use **friend** function
  - Friend function, "**subtract**", which subtracts one **TimeOfDay** object from other and returns a new object of **TimeOfDay** that stores the result.

```
friend TimeOfDay subtract(const TimeOfDay& t1, const TimeOfDay& t2);
// precondition : t1 and t2 have values.
// returns t1 - t2
```

    * Since two **TimeOfDay** objects, we use **friend** function

3.2 Member Function Definitions

3.2.1 constructors

```
TimeOfDay::TimeOfDay(int hours, int minutes):hours(hours),minutes(minutes)
{
    // precondition : hours and minutes are integers.
    // Initializes the time according to the arguments.
    // call checktime() to convert a right form.
    checktime();
}
TimeOfDay::TimeOfDay(int minutes)
:hours(0),minutes(minutes)
{
    checktime();
}
TimeOfDay::TimeOfDay():hours(0),minutes(0)
{
    // Initialize the time to 00:00
}
```

3.2.2 checktime()

```
void TimeOfDay::checktime()
{
    // precondition : hours and minutes are set
    // Modify hours to be between 0 and 23.
    // Modify minutes to be between 0 and 59.
    // * If minutes<0, add 60 to minutes and subtract 1 from hours,
    // repeat those until minutes >=0.
    // * If minutes>59, subtract 60 from minutes and add 1 to hours,
    // repeat those until minutes <= 59.

    while(minutes < 0){
        minutes += 60;
        hours--;
    }
    while(minutes > 59){
        minutes -= 60;
        hours++;
    }
    while(hours < 0){
        hours += 24;
    }
    while(hours > 23){
        hours -= 24;
    }
}
```

3.2.3 accessors

```
int TimeOfDay::get_hours() const
{
// return hours
    return hours;
}
int TimeOfDay::get_minutes() const
{
// return minutes
    return minutes;
}

void TimeOfDay::output(std::ostream &strm) const
{
// Output time to stream
    strm << std::setfill('0') << std::setw(2) << hours << ":"
    << std::setfill('0') << std::setw(2) << minutes << std::endl;
}
```

3.2.4 equal

```
bool equal(const TimeOfDay& t1, const TimeOfDay& t2)
{
// precondition : t1 and t2 have values.
// returns true if t1 and t2 represents the same time;
// otherwise, return false

 //   if ((t1.hours == t2.hours) && (t1.minutes == t2.minutes))return true;
  //  return false;
    return (t1.hours == t2.hours) && (t1.minutes == t2.minutes);
}
```

3.2.5 add

```
TimeOfDay add(const TimeOfDay& t1, const TimeOfDay& t2)
{
// precondition : t1 and t2 have values.
// returns t1 + t2
    TimeOfDay ans(t1.hours + t2.hours,t1.minutes + t2.minutes);
    return ans;
}
```

3.2.6 subtract

```
TimeOfDay subtract(const TimeOfDay& t1, const TimeOfDay& t2)
{
// precondition : t1 and t2 have values.
// returns t1 - t2
    TimeOfDay ans(t1.hours - t2.hours,t1.minutes - t2.minutes);
    return ans;
}
```

3.2.7 main

```
int main(int argc, char **argv)
{
    TimeOfDay time1(1,-9);
    TimeOfDay time2(1,-9);

    time1.output(std::cout);
    time2.output(std::cout);

    if (equal(time1,time2)){
        std::cout << "time1 and time2 are same\n";
    }

    TimeOfDay time3;
    time3 = add(time1,time2);

    time3.output(std::cout);

    TimeOfDay time4;
    time4 = subtract(time3,time2);

    time4.output(std::cout);

    return 0;
}
```

Also look at **sample34.cpp**