Code(Patient Waiting Time)

```
#include <iostream>
#include "Time.h"

using namespace std;

int main()
{

  int numberOfVisits(0), totalWaitingTime(0);
  char answer;
  do {
    bool errorFlag; // arrival, seenByDoctor are instances of Time class
    Time arrival, seenByDoctor;
    cout << "Enter arrival time:";
    arrival.readTime(errorFlag);
    while (errorFlag) {
      cin.clear();
      cin.ignore(INT_MAX, '\n');
      cout << "Arrival time was incorrectly formatted; try again: ";
      arrival.readTime(errorFlag);
    }
    cout << "Enter time seen by doctor:";
    seenByDoctor.readTime(errorFlag);
    while (errorFlag) {
      cin.clear();
      cin.ignore(INT_MAX, '\n');
      cout << "Seen by doctor time was incorrectly formatted; try again: ";
      seenByDoctor.readTime(errorFlag);
    }
    numberOfVisits++;
    // assume that subtracting one Time from another yields the
    // difference in minutes as an int
    totalWaitingTime += seenByDoctor.subtractTimes(arrival);
    cout << "Done? Enter 'y' to quit, anything else to continue: ";
    cin >> answer;
  } while (answer != 'y');
  cout << "Number of visits: " << numberOfVisits << "\n";
  cout << "Total waiting time: " << totalWaitingTime << " minutes.\n";

  cout << "Average wait is " << totalWaitingTime / numberOfVisits
    << " minutes.\n";
  int nAvgTime = totalWaitingTime / numberOfVisits;

  Time arrival_new;
  bool errorFlag;
  cout << "Enter arrival time:";
  arrival_new.readTime(errorFlag);
  while (errorFlag) {
    cin.clear();
    cin.ignore(INT_MAX, '\n');
    cout << "New arrival time was incorrectly formatted; try again: ";
```

```
    arrival_new.readTime(errorFlag);
  }
  int expected_time = arrival_new.AddTime(nAvgTime);
  // cout << expected_time << endl;
  arrival_new.PrintTime();
  return 0;

}
```

```
#pragma once


class Time {
  //
  // Characteristics:
  //
  //   A Time consists of some number of hours and minutes, and is either before noon
  //   (AM) or after noon (PM).
  //
  //   Twelve Noon is 12:00 PM and Twelve Midnight is 12:00 AM.
  //
  //   All Times are assumed to fall on the same day.
  //
public:
  void readTime(bool& errorFlag);
  //   Precondition: Standard input has characters available.
  //   Postconditions: Leading whitespace characters are ignored;
  //      readTime attempts to read, from standard input, a time in
  //      the format <HH>:<MM><A>, where <HH> is an integer between
  //      1 and 12, <MM> is an integer between 0 and 59, and <A> is
  //      either "AM" or "PM". If a properly formatted time can be
  //      read, errorFlag is set to false, and the value of the Time
  //      variable is set to the time read; otherwise, errorFlag is
  //      set to true.
  int subtractTimes(Time t);
  //   Precondition: This Time variable contains a proper value.
  //   Postcondition: None.
  //   Returns: The difference, in minutes, between this Time and Time t.
  //      If this Time occurs prior to Time t, the returned difference
  //      is negative.

  // **** the rest of the class declaration is private
private:
  int minutes;
public:
  int AddTime(int nTime);
  void PrintTime();
};
```

```
#include "Time.h"
#include <iostream>
using namespace std;
```

```
void Time::readTime(bool& errorFlag)
{
  // The time must be formatted as <HH>:<MM><AMorPM>, where
  // <HH> is an int in the range 0 to 12, <MM> is an int in
  // the range 0 to 59, and <AMorPM> is either AM or PM.
  enum AM_PM { AM, PM } AM_or_PM;
  int hour, minute;
  const char delimiter = ':';

  // Assume that the format is bad -- once valid data is extracted,
  // reset errorFlag to false
  errorFlag = true;
  // formatted input -- fail if not an int
  if (!(cin >> hour))
    return;
  if (hour < 0 || hour > 12)
    return;
  char c;
  cin >> c;
  if (c != delimiter)
    return;
  if (!(cin >> minute)) // formatted input
    return;
  if (minute < 0 || minute > 59)
    return;
  cin >> c;
  if (c == 'A' || c == 'a')
    AM_or_PM = AM;
  else if (c == 'P' || c == 'p')
    AM_or_PM = PM;
  else
    return;
  cin >> c;
  if (c != 'M' && c != 'm')
    return;
  errorFlag = false;

  if (hour == 12)
    minutes = minute;
  else
    minutes = hour * 60 + minute;
  if (AM_or_PM == PM)
    minutes += 60 * 12;
}

int Time::subtractTimes(Time t)
{
  return minutes - t.minutes;
}


int Time::AddTime(int nTime)
{
  minutes += nTime;
  return minutes + nTime;
}

void Time::PrintTime()
```

```
{
  int hour(0), minute(minutes);
  enum AM_PM { AM, PM } AM_or_PM;
  hour = minute / 60;
  if (hour > 12)
  {
    hour -= 12;
    AM_or_PM = PM;
  }
  else {
    AM_or_PM = AM;
  }
  minute = minute % 60;
  cout << hour << ":" << minute;
  if (AM_or_PM) {
    cout << "PM";
  }
  else {
    cout << "AM";
  }
}
```