**Composition**

**5th class – Date.h:**

#include <string>

#ifndef DATE\_H

#define DATE\_H

class Date

{

public:

static const unsigned int monthsPerYear{12};

explicit Date(unsigned int=1, unsigned int=1, unsigned int=1900);

std::string toString() const;

~Date();

private:

unsigned int month;

unsigned int day;

unsigned int year;

//utility function in check if day is proper for month and year

unsigned int checkDay(int) const;

};

#endif

**4th class – Date.cpp:**

#include "stdafx.h"

#include "Date.h"

#include <array>

#include <iostream>

#include <sstream>

#include <stdexcept>

Date::Date(unsigned int mn, unsigned int dy, unsigned int yr)

: month{ mn }, day{ checkDay(dy) }, year{ yr } {

if (mn < 1 || mn>monthsPerYear) {

throw std::invalid\_argument("months must be 1-12");

}

// output Date object to show when its constructor is called

std::cout << "Date object constructor for date " << toString() << std::endl;

}

// print Date object in form month/day/year

std::string Date::toString() const {

std::ostringstream output;

output << month << "/" << day << "/" << year;

return output.str();

}

// output Date object to show when its destructor is called

Date::~Date() {

std::cout << "Date object destructor for date " << toString() << std::endl;

}

// utility fucntion to confirm proper day value based on

// month and year; handles leap years, too

unsigned int Date::checkDay(int testDay) const {

static const std::array<int, monthsPerYear + 1> daysPerMonth{

0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31

};

//determine whether testDay is valid for specified month

if (testDay > 0 && testDay <= daysPerMonth[month]) {

return testDay;

}

// February 29 check for leap year

if (month == 2 && testDay == 29 && (year % 400 == 0 ||

(year % 4 == 0 && year % 100 != 0))) {

return testDay;

}

throw std::invalid\_argument("Invalid day for current month and year");

}

**3rd class – Employee.h:**

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

#include <string>

#include "Date.h"

class Employee

{

public:

Employee(const std::string&, const std::string&,

const Date&, const Date&);

std::string toString() const;

~Employee();

private:

std::string firstName;

std::string lastName;

const Date birthDate;

const Date hireDate;

};

#endif

**2nd class – Employee.cpp:**

#include "stdafx.h"

#include <iostream>

#include <sstream>

#include "Employee.h"

#include "Date.h"

// constructor uses member initializer list to pass initializer

// values to constructors of member objects

Employee::Employee(const std::string& first, const std::string& last,

const Date &dateOfBirth, const Date &dateOfHire)

: firstName{first}, lastName{last}, birthDate{dateOfBirth},

hireDate{dateOfHire} {

// output Employee object to show when constructor is called

std::cout << "Employee object constructor: "

<< firstName << ' ' << lastName << std::endl;

}

//print Employee object

std::string Employee::toString() const {

std::ostringstream output;

output << lastName << ", " << firstName << " Hired: "

<< hireDate.toString() << " Birthday: " << birthDate.toString();

return output.str();

}

//output Employee object to show when its destructor is called

Employee::~Employee() {

std::cout << "Employee object destructor: "

<< lastName << ", " << firstName << std::endl;

}

**1st class – ConsoleApplication7.cpp:**

#include "stdafx.h"

#include <iostream>

#include "Date.h"

#include "Employee.h"

int main(){

Date birth{7, 24, 1949};

Date hire{3, 12, 1988};

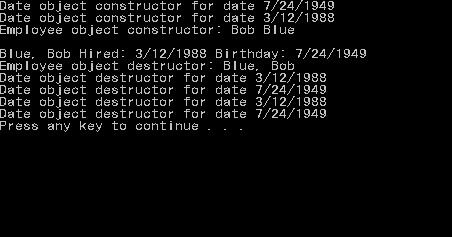
Employee manager{ "Bob", "Blue", birth, hire };

std::cout << "\n" << manager.toString() << std::endl;

return 0;

}

**Result:**



**Important notes:**

* This code composition is only for reading and understanding