CL-217 OBJECT ORIENTED PROGRAMMING LAB

LAB TASK 7

INSTRUCTOR: MUHAMMAD HAMZA

SEMESTER SPRING 2020

The following statements define the class circleType to implement the basic properties of a circle:

```
class circleType
{
public:
void setRadius(double r);
//Function to set the radius.
//Postcondition: if (r >= 0) radius = r;
//
otherwise radius = 0;
double getRadius();
//Function to return the radius.
//Postcondition: The value of radius is returned.
double area();
//Function to return the area of a circle.
//Postcondition: Area is calculated and returned.
double circumference();
//Function to return the circumference of a circle.
//Postcondition: Circumference is calculated and returned.
circleType(double r = 0);
//Constructor with a default parameter.
//Radius is set according to the parameter.
//The default value of the radius is 0.0;
//Postcondition: radius = r;
private:
double radius;
};
```

The definitions of the member functions are as follows:

```
void circleType::setRadius(double r)
{
if (r >= 0)
radius = r;
else
radius = 0;
}

double circleType::getRadius()
{
return radius;
}
double circleType::area()
{
return 3.1416 * radius * radius;
```

```
}
double circleType::circumference()
{
return 2 * 3.1416 * radius;
}
circleType::circleType(double r)
{
setRadius(r);
}
```

Q1. The class circleType provides the basic properties of a circle. (Add the function print to this class to output the radius, area, and circumference of a circle.) Now every cylinder has a base and height, where the base is a circle. Design a class cylinderType that can capture the properties of a cylinder and perform the usual operations on the cylinder. Derive this class from the class circleType. Some of the operations that can be performed on a cylinder are as follows: calculate and print the volume, calculate and print the surface area, set the height, set the radius of the base, and set the center of the base. Also, write a program to test various operations on a cylinder.

Q2. Amanda and Tyler opened a business that specializes in shipping liquids, such as milk, juice, and water, in cylindrical containers. The shipping charges depend on the amount of the liquid in the container. (For simplicity, you may assume that the container is filled to the top.) They also provide the option to paint the outside of the container for a reasonable amount.

Write a program that does the following:

a. Prompts the user to input the dimensions (in feet) of the container (radius of the base and the height).
b. Prompts the user to input the shipping cost per liter.
c. Prompts the user to input the paint cost per square foot. (Assume that the entire container including the top and bottom needs to be painted.)
d. Separately outputs the shipping cost and the cost of painting. Your program must use the class cylinderType to store the radius of the base and the height of the container. (Note that 1 cubic feet 5 28.32 liters or 1 liter 5 0.353146667 cubic feet.)

Using classes, design an online address book to keep track of the names,addresses, phone numbers, and dates of birth of family members, close friends, and certain business associates. Your program should be able tohandle a maximum of 500 entries.

a. Define a class addressType that can store a street address, city, state, and ZIP code. Use the appropriate functions to print and store the address. Also, use constructors to automatically initialize the member variables.

b. Define a class extPersonType using the class personType (implementation given), the class dateType (implementation given), and the class addressType . Add a member variable to this class to classify the person as a family member, friend, or business associate. Also, add a member variable to store the phone number. Add (or override) the functions to print and store the appropriate information. Use constructors to automatically initialize the member variables.
c. Define the class addressBookType using the previously defined classes. An object of the type addressBookType should be able to process a maximum of 500 entries.

The program should perform the following operations:
I. Load the data into the address book from a disk.
II. Sort the address book by last name.
III. Search for a person by last name.
IV. Print the address, phone number, and date of birth (if it exists) of a given person.
V. Print the names of the people whose birthdays are in a given month.
VI. Print the names of all the people between two last names.
VII. Depending on the user's request, print the names of all family members, friends, or business associates.

```
#include <string>
using namespace std;
class personType
{
public:
void print() const;
//Function to output the first name and last name
//in the form firstName lastName.
```

```
    void setName(string first, string last);
      //Function to set firstName and lastName according
      //to the parameters.
      //Postcondition: firstName = first; lastName = last

    string getFirstName() const;
      //Function to return the first name.
      //Postcondition: The value of firstName is returned.

    string getLastName() const;
      //Function to return the last name.
      //Postcondition: The value of lastName is returned.

    personType(string first = "", string last = "");
      //Constructor
      //Sets firstName and lastName according to the parameters.
      //The default values of the parameters are null strings.
      //Postcondition: firstName = first; lastName = last

 private:
    string firstName; //variable to store the first name
    string lastName;  //variable to store the last name
};
```

Figure 10-11 shows the UML class diagram of the `class` **personType**.

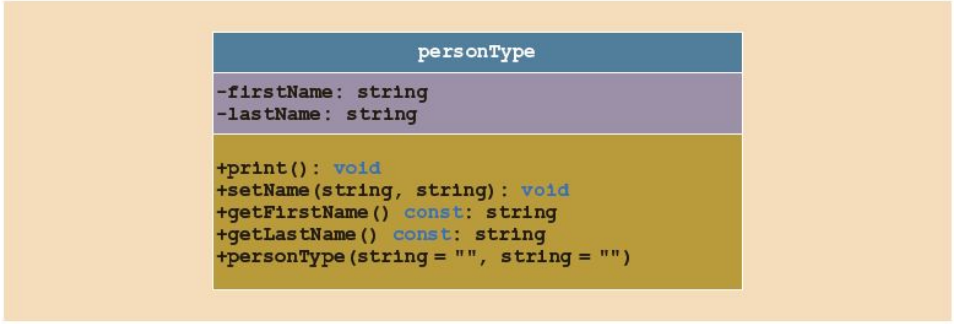

FIGURE 10-11 UML class diagram of the `class` **personType**

We now give the definitions of the member functions of the `class` **personType**.

```
void personType::print() const
{
    cout << firstName << " " << lastName;
}

void personType::setName(string first, string last)
{
    firstName = first;
    lastName = last;
}
```

```
string personType::getFirstName() const
{
return firstName;
}
string personType::getLastName() const
{
return lastName;
}
//constructor
personType::personType(string first, string last)
{
firstName = first;
lastName = last;
}

class dateType
{
public:
void setDate(int month, int day, int year);
//Function to set the date.
//The member variables dMonth, dDay, and dYear are set
//according to the parameters.
//Postcondition: dMonth = month; dDay = day;
//
dYear = year;
int getDay() const;
//Function to return the day.
//Postcondition: The value of dDay is returned.
int getMonth() const;
//Function to return the month.
//Postcondition: The value of dMonth is returned.
int getYear() const;
//Function to return the year.
//Postcondition: The value of dYear is returned.
void printDate() const;
//Function to output the date in the form mm-dd-yyyy.
dateType(int month = 1, int day = 1, int year = 1900);
//Constructor to set the date
//The member variables dMonth, dDay, and dYear are set
//according to the parameters.
//Postcondition: dMonth = month; dDay = day; dYear = year;
//
If no values are specified, the default
```

//
values are used to initialize the member
//
variables.

```
private:
    int dMonth; //variable to store the month
    int dDay;   //variable to store the day
    int dYear;  //variable to store the year
};
```

Figure 11-7 shows the UML class diagram of the `class` **`dateType`**.

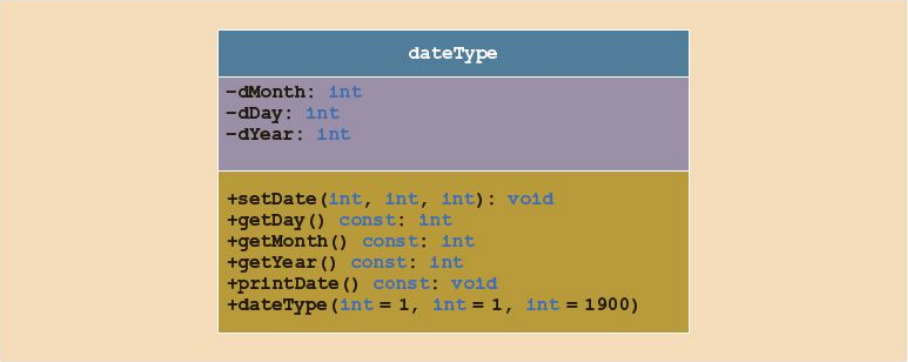

FIGURE 11-7 UML class diagram of the `class` **`dateType`**

The definitions of the member functions of the `class` **`dateType`** are as follows:

```
void dateType::setDate(int month, int day, int year)
{
    dMonth = month;
    dDay = day;
    dYear = year;
}
```

The definition of the function **`setDate`**, before storing the date into the member variables, does not check whether the date is valid. That is, it does not confirm whether **`month`** is between `1` and `12`, **`year`** is greater than `0`, and **`day`** is valid (for example, for January, **`day`** should be between `1` and `31`). In Programming Exercise 2 at the end of this chapter, you are asked to rewrite the definition of the function **`setDate`** so that the date is validated before storing it in the member variables. The definitions of the remaining member functions are as follows:

```
int dateType::getDay() const
{
    return dDay;
}

int dateType::getMonth() const
{
    return dMonth;
}
```



```
int dateType::getYear() const
{
return dYear;
```

```
}
void dateType::printDate() const
{
cout << dMonth << "-" << dDay << "-" << dYear;
}
//Constructor with parameters
dateType::dateType(int month, int day, int year)
{
dMonth = month;
dDay = day;
dYear = year;
}
```