CMP 1001

Introduction to Programming

Part 10: Classes & Objects

by

Assist. Prof. M. Şükrü Kuran

CMP 1001

Section 8: Classes & Objects

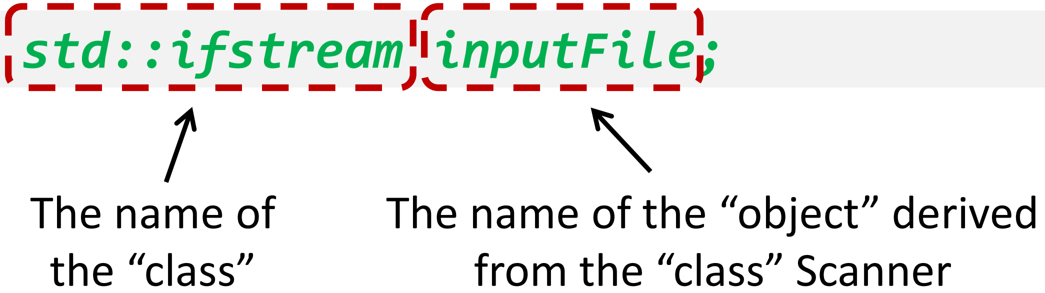
CLASSES & OBJECTS

Complex Variable Types

Up until now, the data types we have been using are mostly the primitive data types (i.e., int, double, char, float, long, boolean, …).

In C++, aside from these primitive data types, we also have more complicated data types and they are called “**classes**”.

When you **DECLARE** a **VARIABLE** of a given Class type, that variable is called an “**object**”.



```
std::ifstream inputFile;
```

The name of the “class”

The name of the “object” derived from the “class” Scanner

LOCAL CLASSES

Defining Local Classes

Classes are comprised of two types of items:

1. Variables (of primitive types or other objects)
2. Functions

The simplest way of defining a class is using a “**Local class**”.

```
class <name of the class>{
    <declaration of the variables & functions of the class>
 };
```

INITIALIZING A CLASS

Defining an Object Based on a Class

Example:

Define a class named “point” which holds the information of a point in a 2D space with both X and Y coordinates as real numbers. Also define another class named “line” which represents a line in a 2D space and is comprised of two point objects of class “point”.

PUBLIC & PRIVATE

Two Main Access Modifiers in C++

A variable or function of a class can be given an **ACCESS MODIFIER** as

public	**private**
Any object can access these variables or functions	Only the methods of that class can access these variables or functions

If you do **NOT** explicitly define an access modifier, the default is set to be "**private**".

NOTE: There is also a third access modifier called "protected" that we will NOT cover in this course.

PUBLIC & PRIVATE

Example

Why do we use these access modifiers?

Basically, to control and limit who can access which variable & function. In a sense for better organization and for better privacy.

Example:

Modify the classes point and line so that the local variables of the class “point” and class “line” are public variables. Also, write a public function for the line class named “length” that calculates and returns the length of the line object.

CONSTRUCTORS

Constructor Functions

A class can have a special function called a “**constructor**”.

A “**constructor**” function **DOES NOT HAVE** a return type, and its name should always be the **NAME OF THE CLASS**.

```
<name of the class> (<parameters>)
```

Constructors should be “**public**”, as they define the main way of initialization of the object.

CONSTRUCTORS

Example

A class can have **MORE THAN ONE** constructor.

In this case each constructor **MUST** have different parameters. This is called "**constructor overloading**"**.**

Example:
Write two constructors to the line class. One that takes 2 points as input another that takes four doubles.

Example:
Using the point class we have defined, define another class called circle which has a point and a double for local variables, as well as two functions named "calculateArea" and "calculateCircumference".

GLOBAL CLASSES

Defining Global Classes

As the name suggests, a "**local class**" is only defined in the file it is declared.

Usually, we want classes to be accessed from other files. Therefore, usually **EACH** class is defined in a separate file. These classes are called "**global classes**".

In case of "**global classes**", the class name and the name of the file **MUST** be the **SAME**.

"**Global classes**" **MUST** be given an access modifier (i.e., public or private).

GLOBAL CLASSES

Header and Source Files of a Class

line2D.h

```
class line2D{
public:
  point2D p1;
  point2D p2;
public:
  double calculateLength();
};
```

Declaration of variables, functions

line2D.cpp

```
#include “line2D.h”

double line2D::calculateLength(){
    ...
}
```

Implementations of variables, functions

GLOBAL CLASSES

Examples

Example:

Define a class named “Employee” with four private variables: the name (String), surname (String), ID number (integer), and age (integer) of the employee.

Write a constructor for the class that takes two Strings and two integers. Also write a void function named printEmployee that prints the ID, name, and surname of the employee.

Define another class named “Company” with three private variables: the name of the company (String), the employees of the company (Employee[]), and the employee count (int).

Write a constructor and two functions named addEmployee and findYoungestEmployee.

Write a program that reads the information of 8 people (i.e., names, surnames, ID numbers, and ages), finds and prints the information of the youngest and oldest employees.

STATIC VARIABLES

Shared Variables Among All Instances of a Class

Local variables of a class can also be defined with the attribute "**static**" when they are declared.

Unlike regular variables, static variables are shared among **ALL INSTANCES** of the class. In other words, a static variable is **THE SAME VARIABLE FOR ALL OBJECTS DERIVED FROM THAT CLASS**.

This means, if the value of the static variable is changed in one object derived from a class, its value has been **CHANGED** in **ALL** objects derived from that class.

STATIC VARIABLES

Example

Static variables can **ONLY** be access through other static functions.

Example:

Add a public static variable named “count” of type int to the “Employee” class. Write a program that reads 4 employees and changes count variable after reading each employee. At the end, print the count variable’s value to the screen.

STATIC FUNCTIONS

Class Functions That Do Not Need Objects

Functions can also be defined with the attribute “**static**” when they are declared.

This means that these functions can be accessed **WITHOUT DECLARING** any object from that class.

Math.sqrt is a static function. We do **NOT** declare an object from class Math when we are using this function.

THE END

Thank you for listening