Object oriented programming: Classes

Comp Sci 1570 Introduction to C++

Missouri S&T | Computer Science

Outline

1. Introduction
   - OOP
2. Classes
   - Definitions
   - Access specifiers
   - Member functions
   - Objects
   - Access

Outline

1. Introduction
   - OOP
2. Classes
   - Definitions
   - Access specifiers
   - Member functions
   - Objects
   - Access

Object oriented programming

- An object is an entity that combines both data and procedures in a single unit. An object's data items, also referred to as its attributes, are stored in member variables. The procedures that an object performs are called its member functions. This wrapping of an object's data and procedures together is called encapsulation.
- Not only objects encapsulate associated data and procedures, they also permit data hiding. Data hiding refers to an object's ability to hide its data from code outside the object. Only the object's member functions can directly access and make changes to the object's data.
- Advantages of Object oriented programming.
  - Software complexity can be easily managed
  - Object-oriented systems can be easily upgraded
  - It is quite easy to partition the work in a project based on object

Outline

1. Introduction
   - OOP
2. Classes
   - Definitions
   - Access specifiers
   - Member functions
   - Objects
   - Access

Outline

1. Introduction
   - OOP
2. Classes
   - Definitions
   - Access specifiers
   - Member functions
   - Objects
   - Access

Classes

The mechanism that allows you to combine data and the function in a single unit is called a class. Once a class is defined, you can declare variables of that type. A class variable is called object or instance. In other words, a class would be the data type, and an object would be the variable. Classes are generally declared using the keyword class, with the following format:

```
class class_name{
  access_specifier_1:
    member1;
  access_specifier_2:
    member2;
  ...
} object_names;
```

Outline

1 Introduction
- OOP

2 Classes
- Definitions
- Access specifiers
- Member functions
- Objects
- Access

Classes are like structs but default to private

```
class class_name{
    private:
        members1;
    protected:
        members2;
    public:
        members3;
};
```

- Members can be either data or function declarations.
- **Private** members of a class are accessible only from within other members of the same class. You cannot access it outside of the class.
- **Protected** members are accessible from members of their same class and also from members of their derived classes.
- **Public** members are accessible from anywhere where the object is visible.

Outline

1 Introduction
- OOP

2 Classes
- Definitions
- Access specifiers
- Member functions
- Objects
- Access

Example: Classes usually have member functions

```
class Circle
{
    private:
        double radius;
    public:
        void setRadius(double r)
        {
            radius = r;
        }
        double getArea()
        {
            return 3.14 * radius * radius;
        }
};
```

Outline

1 Introduction
- OOP

2 Classes
- Definitions
- Access specifiers
- Member functions
- Objects
- Access

Example: create an object of a class

Once a class is defined, you can declare objects of that type. The syntax for declaring a object is the same as that for declaring any other variable. The following statements declare two objects of type circle.

```
Circle c1, c2;
```

Outline

1 Introduction
- OOP

2 Classes
- Definitions
- Access specifiers
- Member functions
- Objects
- Access

Example: Member access

Once an object of a class is declared, it can access the public members of the class.

```
c1.setRadius(2.5);
```

Examples

Check out the code demos