Classes (7.10)

- The reserved word **class** is used to create the complex **structure**
- Classes differ from structures in that:
  - They don't just combine simple data types into one object
  - They also describe how that data can be manipulated

More on Objects

- Object-oriented programming hides the details of objects from objects of other types
- When an object needs information from another object or needs another object to perform a task, it sends a message to the object requesting what it needs
- As a result, object-oriented programs can be written more generically than structured programs
- Usually, making changes to the object-oriented programs is easier than changing structured programs

class Format

- A **class** is like a **struct** but much more
- Whereas **structs** can contain simple data types, **classes** contain both *data types* and *functions* that manipulate the class data

```
class ClassName
{
  Declarations for member variables
  Declarations for member functions
};
```

A C++ Example

- Enough of theory!
- Let's have a look at a real example.
- We will create a **`class Person`** that will:
  - Store information about person
  - Store functions to manipulate this information

The Person Class

```
class Person
{
public:
  int age;

  int returnAge();
  int returnBirthYear();
};
int main()
{
  Person cPerson;
  cPerson.age = 28;
  cout << "person is: " << cPerson.returnAge();
  cout << "person was born in: "
       << cPerson.returnBirthYear();
  return 0;
}
```

Data Member

Member Function prototypes

Object of class Person

The Person Class

```
int Person::returnAge()
{
  return age;
}

int Person::returnBirthYear()
{
  return 2009 - age;
}
```

Private & Public

- Class data members and member functions can be either private or public
- Private data members and member functions can only be accessed within that class
- Public data members and member functions can be accessed from outside of that class

Example Using Private & Public

```
class Person
{
private:
   int age;
public:
   void setAge(int);
   int returnAge();
   int returnBirthYear();
};
int main()
{
   Person cPerson;
   cPerson.setAge(28);
   cout << "person is: " << cPerson.returnAge() << endl;
   cout << "person was born in: "
        << cPerson.returnBirthYear();
   return 0;
}
```

- Because `age` is a private data member, we can't use `person.age = 28` here.
- Instead, we need to create a new function in the class to set the age.

Continued

```
void Person::setAge(int newAge)
{
   age = newAge;
}
int Person::returnAge()
{
   return age;
}
int Person::returnBirthYear()
{
   return 2009 - age;
}
```

Mutators and Accessors

- A ***mutator*** is any method that changes the value of a member variable

```
void Person::setAge(int newAge)
{
  age = newAge;
}
```

- An ***accessor*** is a method that uses a class member but does not change its value

```
int Person::returnAge()
{
  return age;
}
```


Time class

```
class Time
{
  private:
    int hour;     // 0 - 23 (24-hour clock format)
    int minute;   // 0 - 59
    int second;   // 0 - 59

  public:
    void setTime(int h, int m, int s);
    void printUniversal();              // 13:27:06
    void printStandard();               // 1:27:06 PM

}; // end class Time
```


class questions

- Q1: How many ***members*** does class time have? List them.
- Q2: How many ***methods*** does class time have? List them.
- Q3: How many ***mutators*** does class time have? List them.
- Q4: How many ***accessors*** does class time have? List them.
- P1: Write the definitions of the member functions?
- Q5: Where would they be written?
- Q6: How do we create objects of the class Time?
  - A regular object
  - An array of objects
- P2: Write C++ code that shows how you would use the objects to call the member functions?