CLASSES AND OBJECTS

A class is like a cookie cutter.

An object is like a cookie.

Delroy A. Brinkerhoff

THE OBJECT-ORIENTED PARADIGM

- "*Object-oriented modeling and design* is a way of thinking about problems using models organized around real-world concepts. The fundamental construct is the object, which combines both data structure and behavior in a single entity" (Rumbaugh et al., 1991, p. 1).
- A way of looking at software problems
- A way of implementing software solutions to those problems
- Combining "both data structure and behavior in a single entity" is called *encapsulation*

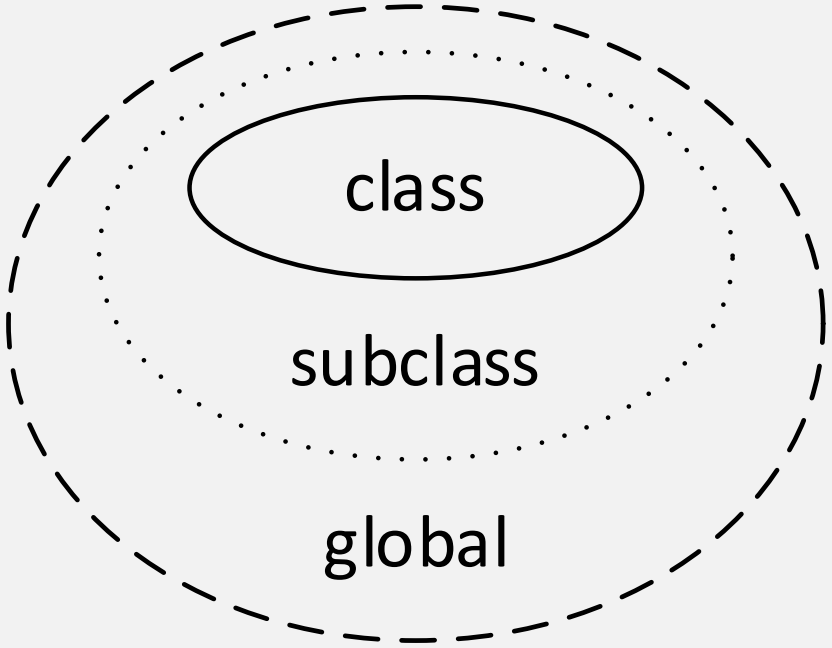
OBJECTS HAVE DATA STRUCTURE

- In the object-oriented model and in the Unified Modeling Language, data structure is known as *attributes*
- In Java, data structure is represented by *instance variables* or *instance fields*
- In C++, data structure is represented by *member variables* or *member data*

OBJECTS HAVE BEHAVIOR

- In the object-oriented model and in the Unified Modeling Language, behavior is known as *operations* or *behaviors*
- In Java, operations are represented by *methods*
- In C++, operations are represented by *member functions*
- Sending a message to an object is a way of invoking one of the object's operations behaviors; which is equivalent to calling one of an objects methods or member functions

FEATURE VISIBILITY OR ACCESSIBILITY



- Collectively, attributes and operations are called features
- Keywords control where features are visible or accessible:
  - private: class only
  - protected: class and subclasses
  - public: everywhere

C++ CLASS SPECIFICATION

A class is a new data type

```
class Time
{
    private:
        int hours;
        int minutes;
        int seconds;

    public:
        Time();
        Time(int h, int m, int s);
        Time(int s);

        Time   add(Time t2);
        Time*  add(Time* t2);

        void   print();
        void   read();
};
```

ACCESSING AN OBJECT'S FEATURES

AUTOMATIC OBJECTS

```
Time start;
Time end;

start.hours = 60;
cout << start.minutes << out;

end.read();
Time total = start.add(end);
total.print();
```

DYNAMIC OBJECTS

```
Time* start = new Time;
Time* end = new Time;

start->hours = 60;
cout << start->minutes << out;

end->read();
Time* total = start->add(end);
total->print();
```