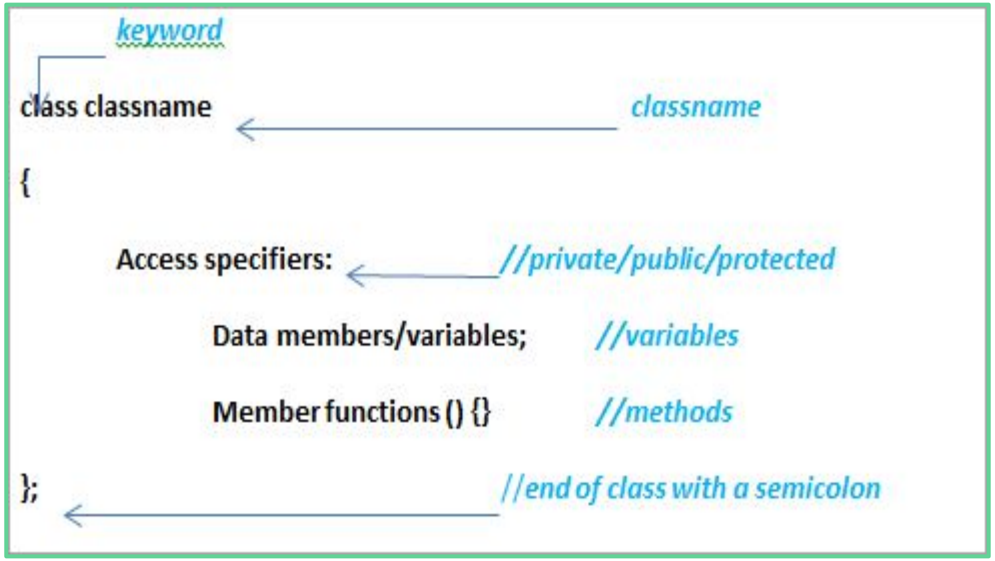
Classes, Enums, and Movies

Lab 10: https://maryash.github.io/135/labs/lab_10.html

- Sadab Hafiz

Introduction to Classes

Classes in C++ are user-defined compound datatypes that can be used for grouping together several related variables and functions. These ‘variables’ inside a class are called fields (or data members) of the classes.



```
class classname
{
        Access specifiers:           //private/public/protected
                Data members/variables;     //variables
                Member functions () {}      //methods
};                                   //end of class with a semicolon
```

Variables of of these class types are referred to as objects. In other words, If you define an instance of your class, you are creating an object of that particular class. Data members and member functions can be private, public or protected depending on the access specifier.

Access Specifiers

Access Specifiers are used to specify who can access certain data members and member functions of a class:

public: Can be accessed by anyone who has access to an object of this class. Usually there are getters and setter functions which allows users to modify private fields. More things are usually public including constructors, destructors etc. that are covered in CS235.

private: Can be accessed ONLY by the class itself. Usually sensitive data members and functions are specified as private. Most classes have their data members as private.

protected: Can be accessed by classes that inherit from the current class. This will be covered more in CS235 and maybe later this semester.

Class to represent Time in HH:MM format

```
// Time class
class Time {
      public:                     // access specifier
            int h;                // hour
            int m;                // minute
};
// Using time class
int main() {
      Time now;                   // empty Time variable
      now.h = 17;                 // assign hour field
      now.m = 45;                 // assign minutes field
}
```

This is a class to represent Time. It has two public data members:

- h: represents hour
- m: represents minutes

The main function creates an empty Time object named now and it updates the h and m fields of this object. The object stores 17:45.

Assignment operator

Instead of creating a new object and updating the fields like the main function in last slide, we can directly create the object with some values. We can also create a new object and make a copy of it from an older object using the assignment operator (=).

```
. . . Time class definition is above the main function

int main() {

      Time morningLecture = {8, 10};

      Time myAlarm;                                         // make another variable

      myAlarm = morningLecture;                             // copying

      cout << morningLecture.h << endl;                     // will print 8

      cout << myAlarm.h << endl;                            // will print 8 as well

}
```

The default assignment operator works because we have a simple class. In your future courses, you will use more complex classes that will require you to **overload** the default operators. Learn more here if you're interested

Functions with object parameters

You can define functions that take objects as parameters. Here’s a useful one:

```
// function to print a time object in correct format

void printTime(Time time) {

    cout << time.h << ":" << time.m;                          // use . to access fields

}

int main() {

      Time morningLecture = {8, 10};

      Time myAlarm;                                             // make another variable

      myAlarm = morningLecture;                                 // copying

      printTime(morningLecture);                                // will print 8:10

      cout << endl;                                             // go to next line

      printTime(myAlarm);                                       // will print 8:10 as well

}
```

Introducing Enum Types

Now that we are a bit more familiar with classes, we can talk about another similar grouping concept known as **Enum**.

```
enum Genre {ACTION, COMEDY, DRAMA, ROMANCE, THRILLER};
```

Enum types work as sets of named values or labels. A variable of type Genre can assume any of the values listed in the curly braces, example:

```
Genre myFavorite = COMEDY;
```

Just like classes, enums can be used as function parameters, data members and variable types. We will create a class in the next slide using Genre enum as a data member.

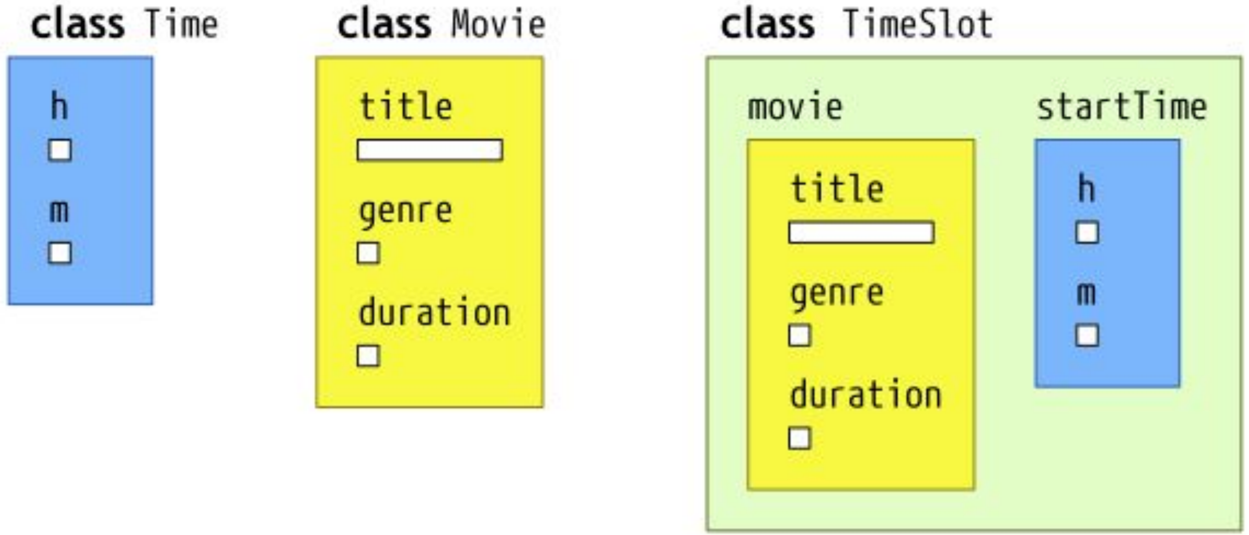
Classes to Represent a Movies and Time slots

Using classes and enums as datatypes for class fields:

```
class Movie {

      public:

            string title;

            Genre genre;                                // only one genre per movie

            int duration;                               // in minutes

};


class TimeSlot {

      public:

            Movie movie;                                // what movie

            Time startTime;                             // when it starts

};
```

Usage example

```
void printMovie(Movie mv){                                                  // function to print a movie object
        string g;
                switch (mv.genre) {                                         // switch statements are similar to if statements
                        case ACTION   : g = "ACTION"; break;
                        case COMEDY   : g = "COMEDY"; break;
                        case DRAMA    : g = "DRAMA";  break;
                        case ROMANCE  : g = "ROMANCE"; break;
                        case THRILLER : g = "THRILLER"; break;
        }
        cout << mv.title << " " << g << " (" << mv.duration << " min)";
}
int main(){
        Movie movie1 = {"Back to the Future", COMEDY, 116};                // create a Movie object
        printMovie(movie1);                                                 // would print “Back to the Future COMEDY (116 min)”
        TimeSlot morning = {movie1, {9, 15}};                               // create a TimeSlot object
}
```


class Time
h
m
class Movie
title
genre
duration
class TimeSlot
movie
title
genre
duration
startTime
h
m