Introduction to object-oriented programming

“OOP”

Classes are custom-defined datatypes

```
class Tree
{
    float age;
    float height;
    int numberOfLeaves;
}
```

You can define objects of a class much the same as you define standard datatypes

```
int a = 7;
float x;

Tree oak1;
Tree oak2;
Tree maple1;
```

Once defined, you can manipulate object data

```
Tree oak1;
oak1.height = 15.6;

Tree maple1;
maple1.height = 42.6;
```

- Each data member is distinct for each object
- (Public) data members can be accessed using `<object>.<datum>`

The real usefulness of classes is that *functionality* can be defined for the class…

```
class Tree
{
    int age;
    //...
    void Grow(void) {age++;}
};
```

… and used by the objects

```
Tree oak1; oak1.age = 15;
Tree maple1; maple1.age = 42;

maple1.Grow(); //this will cause maple1 to grow,
               //but leave oak1 unchanged
```

Even better, those functions can be made to take *arguments*

```
class Tree
{
      int age;
      //...
      int Grow(int years);
};

int Tree::Grow(years)
{
      age+=years;
      return age;
}
```

By making data members private, they can be hidden from the rest of the program

```
class Tree
{
    private:
        int age;
    public:
        int Grow(void);
};
```

A constructor is a special function that is called when you define an object

```
class Tree
{
public:
     Tree(int a) {age = a;}
};

Tree oak1(42);
```

OK, so how is this all useful to you?

- Robot Mesh Studio encapsulates physical components into classes
- Functionality for each class is defined for specific objects
  - `motor`
  - `sonar`
  - `brain`
  - `etc.`

The API defines what each function does

- (Well, sort of...it’s not their best work...)