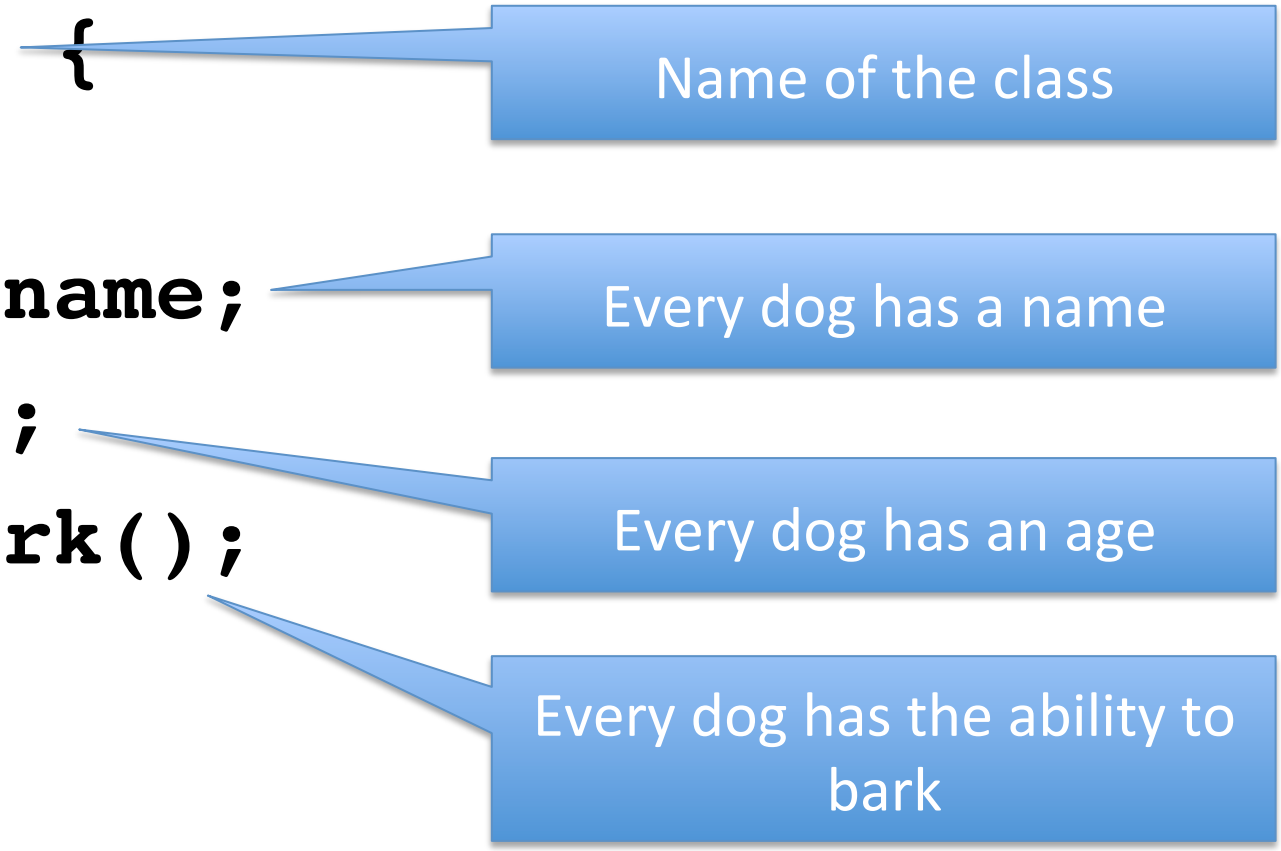
Objects II

- A ***class*** is a struct plus some associated functions that act upon variables of that struct type.
  - class = struct + functions
- An ***object*** is a variable of some struct type
  - aka "an ***instance*** of a class."
- In a class, the variables of that class are called ***fields***; the functions are called ***methods.***
  - Together, the fields and methods are called data members (book uses ***data members*** and ***member functions***).

```
class dog {
  public:
  string name;
  int age;
  void bark();
};

void dog::bark() {
  cout << name << "says woof!";
}
```



```
class dog {
   public:
   string name;
   int age;
   void bark();
};

void dog::bark() {
   cout << name << "says woof!";
}
```

A class's methods are allowed to use the fields defined within that class as local variables.

A method (normally) only has access to the fields for its own object.

```
void dog::bark() {
   cout << name << "says woof!";
}
main
dog regan;
regan.name = "Regan";
regan.age = 3;

dog jack;
jack.name = "Jack";
jack.age = 8;

regan.bark();
jack.bark();
```

regan:

name: "Regan"
age: 3

jack:

name: "Jack"
age: 8

When regan.bark() is called, in the dog::bark() function, name is automatically set to "Regan" and age is set to 3.

```
void dog::bark() {
   cout << name << "says woof!";
}
main
dog regan;
regan.name = "Regan";
regan.age = 3;

dog jack;
jack.name = "Jack";
jack.age = 8;

regan.bark();
jack.bark();
```

regan:

name: "Regan"
age: 3

jack:

name: "Jack"
age: 8

When jack.bark() is called, in the dog::bark() function, name is automatically set to "Jack" and age is set to 8.

- Most object-oriented (OO) programming languages allow us to specify fields and methods as ***public*** or ***private***.
- ***Private*** members can be used only by the person writing the class (i.e., inside methods).
- ***Public*** members can be used by the person writing the class, or the person using the class.

```
class A {
  public:
  int x;
  void f();

private:
  int y;
  void g();
}
```

```
int main()
{
  A obj1, obj2;

  obj1.x = 4;  // ok
  obj1.y = 2;  // error

  obj2.f();    // ok
  obj2.g();    // error
}
```

Why have public and private?

- Sometimes we need to ***hide*** certain variables or functions from the user of a class so the user doesn't accidentally screw things up.
- This is called ***information hiding***.
- Used to protect the members of an object that should only be used by the person writing the class.

```
class dog {
   public:
   string name;
   int age;
   void bark();
};

void dog::bark() {
   cout << name << "says woof!";
}
```

What could go wrong with age or name being public?

```
class dog {
  public:
  void bark();

  private:
  string name;
  int age;
};

void dog::bark() {
  cout << name << "says woof!";
}
```

Good rule of thumb to make all fields (variables) private unless you have a very good reason not to.

main

```
dog regan;
regan.name = "Regan";
regan.age = 3;

dog jack;
jack.name = "Jack";
jack.age = 8;

regan.bark();
jack.bark();
cout << "Jack is " << jack.age << endl;
```

What is wrong with this code now?

main

```
dog regan;
regan.name = "Regan";
regan.age = 3;

dog jack;
jack.name = "Jack";
jack.age = 8;

regan.bark();
jack.bark();
cout << "Jack is " << jack.age << endl;
```

What is wrong with this code now?

Red fields are private; cannot be used outside of the class now.

```
class dog {
  public:
  void bark();
  void setName(string newName);
  string getName();
  void setAge(int newAge);
  int getAge();

  private:
  string name;
  int age;
};  // rest of code on computer
```

Add setters and getters.

- The public members of a class are known as the class's ***interface***.
  - These members are what the users of your class see.
  - Generally describes ***what*** a class does.
- The private members of a class are known as the class's ***implementation***.
  - These are hidden from the user.
  - Generally describe how a class works.
- We strive to keep a class's interface consistent over time. We can change the implementation any time we want.

What is in a car's interface and implementation?

```
class dog {
  public:
  int getAgeDogYears();
  int getAgeHumanYears();
  void setAgeDogYears(int newAge);
  void setAgeHumanYears(int newAge);

  private:
  // What would this look like?
};
```

- To your dog class, add the ability for the dog to have some amount of energy. ***The dog's energy can never go below zero.***
- Add a method called setEnergy(int newEnergy) so the user can set the starting energy for the dog.
- Add a method for the dog to playFetch(). Playing fetch tires the dog out, so it lowers the dogs energy by some amount.
- Add a method for the dog to sleep for a certain number of hours. The dog's energy should be raised proportionally to the number of hours it sleeps.
- Extra: add more methods that change the dog's energy in some way. Add more fields and methods.