Classes

Jack Garner

September 18, 2019

h

C++ uses two kinds of files, h files and cpp files. H files are just like cpp files, but, by convention, contain definitions, not instances.

Can be included with

#include "fileName.h"

classes

h file:

```
class Test {
  private:
  int x;
  int y;
  public:
  Test();
  int getX();
  int getY();
  int z;
  static int z;
};
```

classes

cpp file:

```cpp
Test::Test() {
        std::cout << "In constructor" << std::endl;
}
Test::getX() {
        return x;
}
Test::getY() {
        return y;
}
int Test::z = 5;

Test t;
Test t2 = Test();
Test t2 = new Test(); // Make it on the heap
// Test t(5); if Test took parameters;
```

Inheritance

```cpp
class A {
  private:
  public:
  virtual void f() = 0;
  virtual void ff() { cout << "test" << endl; };
};
class B : A {
  private:
  public:
  virtual void f() override { cout << "over" << endl; };
};
```

What’s different?

- Classes are split between definitions and implementations
- public and private define block of members
- Static things exist
- Inheritance looks different
- Virtual vs pure vs not virtual

What’s the same?

- Classes are created in a similar way
- Same concepts
- Has inheritance

Try It!

Create a base class for Animals which says a default name on construction. Create two new animals extending the Animal class (like dog and cat). Each new animal should have a constructor which says its name. Create a combination animal by inheriting from both of the animals. (For example, class DogCat : Dog, Cat {...) Create an instance of the animal and see what it does.