Inheritance and Streams

We say class B is derived from class A if:

• Class B is a more specific type of Class A
• Both classes have some things in common
• Class B has more features
Inheritance and Streams

Examples from the real-world:

- **Cat** has a *derived from* relationship with **Animal**
- **Car** has a *derived from* relationship with **Vehicle**
- **Electrical Engineering** has a *derived from* relationship with **Engineering**

<table>
<thead>
<tr>
<th>More specific</th>
<th>More general</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>Derived From Animal</td>
</tr>
<tr>
<td>Car</td>
<td>Derived From Vehicle</td>
</tr>
<tr>
<td>Electrical Engineering</td>
<td>Derived From Engineering</td>
</tr>
</tbody>
</table>

![Child](#) — **Parent**
Inheritance and Streams

C++ example #1

```cpp
ifstream istream
```

- **Derived From**
- **ifstream** is for files
- **istream** is for input streams (e.g. `cin` is an object of type `istream`)
- Shared behavior: can use `>>` on both
- Behavior specific to `ifstream`: can use `open()` and `close()` on `ifstream` but not on `istream`

---

We say `ifstream` is derived from `istream`.
- An `ifstream` object has all the properties of an `istream` object.
- An `ifstream` object is also an object of type `istream`.
Inheritance and Streams

C++ example #2

ofstream Derived From ostream

Child Parent

- ofstream is for files
- ostream is for output streams (e.g., cout and cerr are objects of type ostream)
- Shared behavior: can use << on both
- Behavior specific to ofstream: can use open() and close() on ofstream but not on ostream

Inheritance and Streams

C++ example #2

ofstream Derived From ostream

Child Parent

- We say ofstream is derived from ostream
- An ofstream object has all the properties of an ostream object
- An ofstream object is also an object of type ostream
Inheritance and Streams

```cpp
void foo1(ifstream& sourceFile)
{
    int n1, n2;
    sourceFile >> n1 >> n2;
    cout << n1 << " + " << n2 << " = " << (n1 + n2) << endl;
}

ifstream fin;
fin.open("input.txt");
foo1(fin); /* Works */
foo1(cin); /* Doesn’t work because cin is not an ifstream */
```

Inheritance and Streams

```cpp
void foo2(istream& sourceFile)
{
    int n1, n2;
    sourceFile >> n1 >> n2;
    cout << n1 << " + " << n2 << " = " << (n1 + n2) << endl;
}

ifstream fin;
fin.open("input.txt");
foo2(fin); /* Works */
foo2(cin); /* Works */
```
The stringstream class

stringstream

• This is a handy class for creating strings
• stringstream derived from iostream which is derived from istream
```cpp
#include <sstream>
#include <string>
#include <iostream>

int main(int argc, char** argv) {
    std::stringstream ss;
    ss.clear();
    ss.str("1. ");

    float c = 9.99;
    ss << "Macho burrito: $" << c;
    std::cout << ss.str() << std::endl;
    return 0;
}
```

Must include this header to use stringstream

```cpp
#include <sstream>
#include <string>
#include <iostream>

int main(int argc, char** argv) {
    std::stringstream ss;
    ss.clear();
    ss.str("1. ");

    float c = 9.99;
    ss << "Macho burrito: $" << c;
    std::cout << ss.str() << std::endl;
    return 0;
}
```
#include <sstream>
#include <string>
#include <iostream>

int main(int argc, char** argv) {
    std::stringstream ss;
    ss.clear();
    ss.str("1. ");

    float c = 9.99;
    ss << "Macho burrito: $" << c;
    std::cout << ss.str() << std::endl;
    return 0;
}
The code in the previous slides prints out:
1. Macho burrito: $9.99

You can also use stringstreams to parse strings (see next slide)
```cpp
#include <sstream>
#include <string>
#include <iostream>

int main(int argc, char** argv) {
    int s1, s2, s3;
    std::stringstream ss("100 80 50");
    ss >> s1;
    ss >> s2;
    ss >> s3;
    std::cout << (s1+s2+s3)/3.0 << std::endl;
    return 0;
}
```
The code in the previous slide prints out:
76.6667