

Polymorphism

Object-Orientated Programming (OOP) concept that a single object may take on the type of any of its base types.

- A Planet may polymorph itself to a Sphere
- A Sphere cannot polymorph to be a Planet (*base types only*)

Virtual

- The **virtual** keyword allows us to override the behavior of a class by its derived type.

Sphere.cpp	Planet.cpp
<pre>Sphere::print_1() { cout << "Sphere" << endl; } Sphere::print_2() { cout << "Sphere" << endl; } virtual Sphere::print_3() { cout << "Sphere" << endl; } virtual Sphere::print_4() { cout << "Sphere" << endl; } // .h: virtual Sphere::print_5() = 0;</pre>	<pre>// No print_1() defined // in Planet Planet::print_2() { cout << "Earth" << endl; } // No print_3() defined // in Planet Planet::print_4() { cout << "Earth" << endl; } Planet::print_5() { cout << "Earth" << endl; }</pre>

	Sphere obj;	Planet obj;	Planet r; Sphere &obj = r;
obj.print_1();			
obj.print_2();			
obj.print_3();			
obj.print_4();			
obj.print_5();			

Why?

Suppose you're managing an animal shelter that adopts cats and dogs.

Option 1 – No Inheritance

animalShelter.cpp	
1	Cat & AnimalShelter::adopt() { ... }
2	Dog & AnimalShelter::adopt() { ... }
3	...

Option 2 – Inheritance

animalShelter.cpp	
1	Animal & AnimalShelter::adopt() { ... }

Pure Virtual Methods

In Sphere, print_5() is a **pure virtual** method:

Sphere.h	
1	virtual Sphere::print_5() = 0;

A pure virtual method does not have a definition and makes the class and **abstract class**.

Abstract Class:

1. [Requirement]:
2. [Syntax]:
3. [As a result]:

Abstract Class Animal

In our animal shelter, **Animal** is an abstract class:

Assignment Operator

I didn't cover a few details of the assignment operator -- let's do that:

1. [Default Assignment Operator]

C++ generate a default assignment operator for simple classes:

- No non-static const variables
- No reference variables

2. [Self-Assignment]

- Programmers are never perfect and are never optimal. Consider the following:

```
assignmentOpSelf.cpp
1 #include "Sphere.h"
2
3 int main() {
4     cs225::Sphere s(10);
5     s = s;
6     return 0;
7 }
```

- Ensure your assignment operator doesn't self-destroy:

```
assignmentOpSelf.cpp
1 #include "Sphere.h"
2
3 Sphere& Sphere::operator=(const Sphere &other) {
4     if (&other != this) {
5         _destroy();
6         _copy(other);
7     }
8     return *this;
9 }
```

Abstract Data Types (ADT):

List ADT - Purpose	Function Definition

List Implementation

What types of List do we want?

Templated Functions:

```
functionTemplatel.cpp
1
2
3 T maximum(T a, T b) {
4     T result;
5     result = (a > b) ? a : b;
6     return result;
7 }
```

Templated Classes:

```
List.h
1 #ifndef LIST_H_
2 #define LIST_H_
3
4
5
6 class List {
7     public:
8
9
10
11     private:
12
13
14 };
15
16 #endif
```

```
List.cpp
1
2
3
4
5
```

CS 225 – Things To Be Doing:

1. Theory Exam #1 – Today's the final day of the exam.
2. MP2 due Jan. 12 (10 days), EC deadline in 3 days!
3. Lab Extra Credit → Attendance in your registered lab section!
4. Daily POTDs