Destructor
The last and final member function called in the lifecycle of a class is the destructor.

Purpose of a destructor:

The automatic destructor:
1. Like a constructor and copy constructor, an automatic destructor exists only when no custom destructor is defined.
2. [Invoked]:
3. [Functionality]:

Custom Destructor:

```cpp
…necessary if you need to delete any heap memory!
```

Overloading Operators
C++ allows custom behaviors to be defined on over 20 operators:

```
<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>+</th>
<th>-</th>
<th>*</th>
<th>/</th>
<th>%</th>
<th>++</th>
<th>--</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitwise</td>
<td>&amp;</td>
<td></td>
<td>^</td>
<td>~</td>
<td>&lt;&lt;</td>
<td>&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td>Assignment</td>
<td>=</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comparison</td>
<td>==</td>
<td>!=</td>
<td>&gt;</td>
<td>&lt;</td>
<td>&gt;=</td>
<td>&lt;=</td>
<td></td>
</tr>
<tr>
<td>Logical</td>
<td>!</td>
<td>&amp;</td>
<td>&amp;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>[ ]</td>
<td>()</td>
<td>-&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

General Syntax:
Inheritance
In nearly all object-oriented languages (including C++), classes can be extended to build other classes. We call the class being extended the base class and the class inheriting the functionality the derived class.

Derived Class: Planet

```cpp
#ifndef PLANET_H_
#define PLANET_H_
#include "Sphere.h"
class Planet : public cs225::Sphere {
    // Empty!
};
#endif
```

In the above code, Planet is derived from the base class Sphere:
- All public functionality of Sphere is part of Planet:

```
int main() {
    Planet p;
    p.getRadius(); // Returns 1, the radius init'd // by Sphere's default ctor
    ...}
```
- [Private Members of Sphere]:

Adding New Functionality:

```cpp
class Planet : public cs225::Sphere {
public:
    Planet(std::string name, double radius); // ctor
private:
    std::string name_;}
```

Functions: non-virtual, virtual, and pure virtual
- The virtual keyword:

```
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;
```

```
Planet::print_2() {
    cout << "Earth" << endl;
}
virtual Planet::print_3() {
    cout << "Earth" << endl;
}
virtual Planet::print_4() {
    cout << "Earth" << endl;
}
Planet::print_5() {
    cout << "Earth" << endl;
}
```

CS 225 – Things To Be Doing:
1. Theory Exam #1 is ongoing – ensure you’re signed up!
2. Attend and complete lab_memory (due Sunday)
3. MP2 is ongoing (extra credit due Monday)
4. Daily POTDs every M-F for daily extra credit!