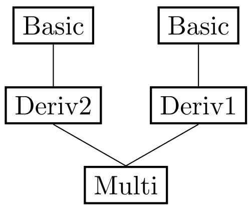
Week 4

Exercise 30

- Draw `Multi`'s class hierarchy

Below is the class hierarchy of `Multi` at this point of the assignment.



- Explain the compiler's error message after the addition of the `static_cast`

As can be seen from the illustration above, due to the way that `Deriv1` and `Deriv2` are constructed, `Basic` is included twice by the time `Multi` inherits from `Deriv1` and `Deriv2`. Hence, the compiler indicates that it does not know which `Basic` to cast to.

- Change the statement so that there is no compilation error

First, the cast can be done in a two-step fashion: first to either `Deriv1` or `Deriv2`, and then to the associated `Basic`. This is achieved as follows:

```
Multi::Multi()
{
  cout << static_cast<Basic *>(static_cast<Deriv1 *>(this)) << '\n';
}
```

Secondly, a `reinterpret_cast` can be used instead, as follows. Note that this is a very dangerous practice, and should be used with extreme caution.

```
Multi::Multi()
{
  cout << reinterpret_cast<Basic *>(this) << '\n';
}
```

- Show the required modifications to allow the compiler to compile the statement without errors

The best way to solve the compilation error without altering the statement would be to make use of virtual inheritance. As such, the class declaration of `Deriv1` and `Deriv2` should be changed as follows:

```
...
class Deriv1: public virtual Basic
{
};
...
```

```
...
class Deriv2: public virtual Basic
{
};
...
```