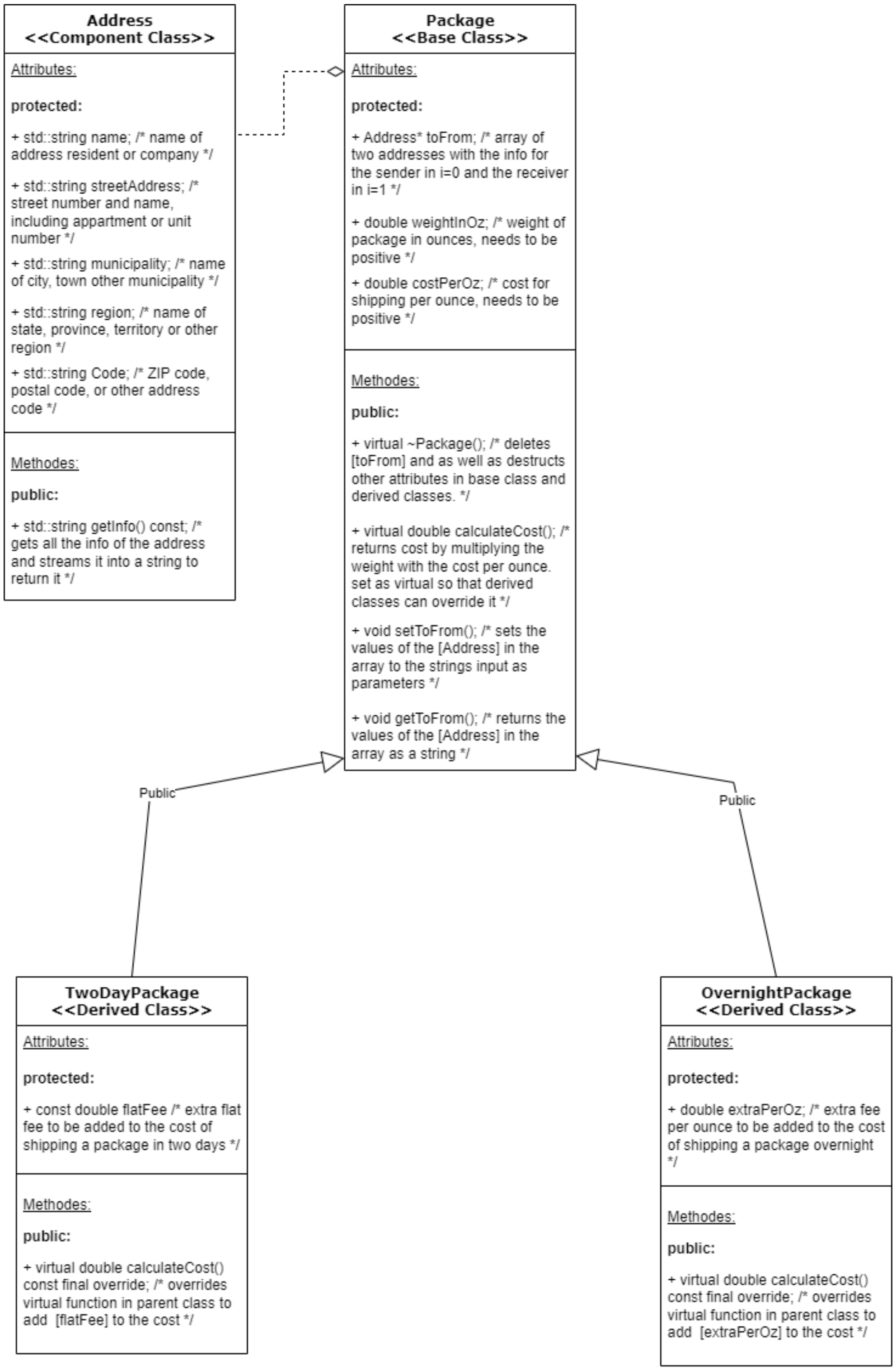
COEN-244 Assignment 3 Report

Romain Giroux (40177867)

Task 1.1



Address
<<Component Class>>

Attributes:

protected:

+ std::string name; /* name of address resident or company */

+ std::string streetAddress; /* street number and name, including appartment or unit number */

+ std::string municipality; /* name of city, town other municipality */

+ std::string region; /* name of state, province, territory or other region */

+ std::string Code; /* ZIP code, postal code, or other address code */

Methodes:

public:

+ std::string getInfo() const; /* gets all the info of the address and streams it into a string to return it */

Package
<<Base Class>>

Attributes:

protected:

+ Address* toFrom; /* array of two addresses with the info for the sender in i=0 and the receiver in i=1 */

+ double weightInOz; /* weight of package in ounces, needs to be positive */

+ double costPerOz; /* cost for shipping per ounce, needs to be positive */

Methodes:

public:

+ virtual ~Package(); /* deletes [toFrom] and as well as destructs other attributes in base class and derived classes. */

+ virtual double calculateCost(); /* returns cost by multiplying the weight with the cost per ounce. set as virtual so that derived classes can override it */

+ void setToFrom(); /* sets the values of the [Address] in the array to the strings input as parameters */

+ void getToFrom(); /* returns the values of the [Address] in the array as a string */

TwoDayPackage
<<Derived Class>>

Attributes:

protected:

+ const double flatFee /* extra flat fee to be added to the cost of shipping a package in two days */

Methodes:

public:

+ virtual double calculateCost() const final override; /* overrides virtual function in parent class to add [flatFee] to the cost */

OvernightPackage
<<Derived Class>>

Attributes:

protected:

+ double extraPerOz; /* extra fee per ounce to be added to the cost of shipping a package overnight */

Methodes:

public:

+ virtual double calculateCost() const final override; /* overrides virtual function in parent class to add [extraPerOz] to the cost */

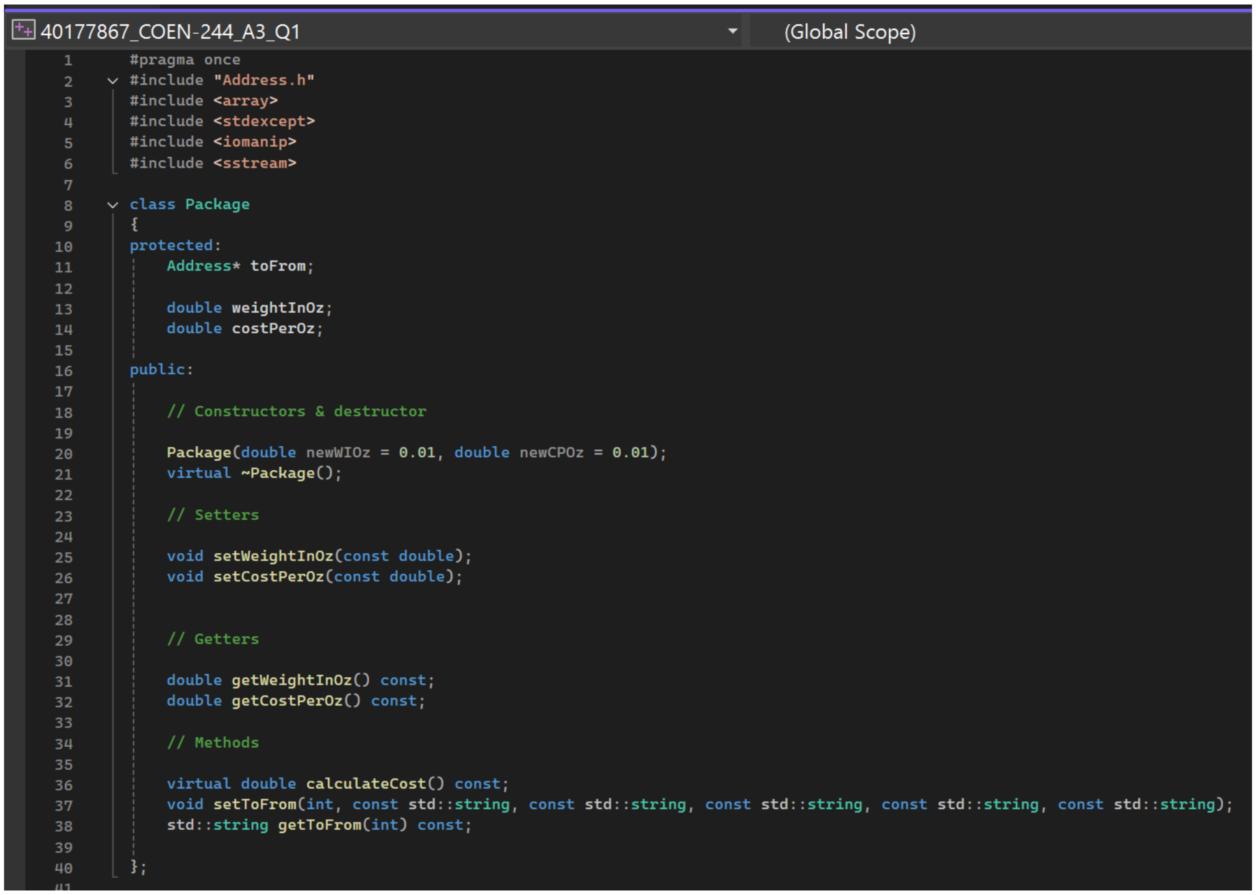
Fig.1: UML diagram for question 1.

Designing a UML diagram for a program is often an underappreciated step in the software development process for most newcomers to the subject such as myself. It can feel like an unnecessary step, why not just start coding right away and design as you go? Why should we let such task stop us from starting the fun part right away?

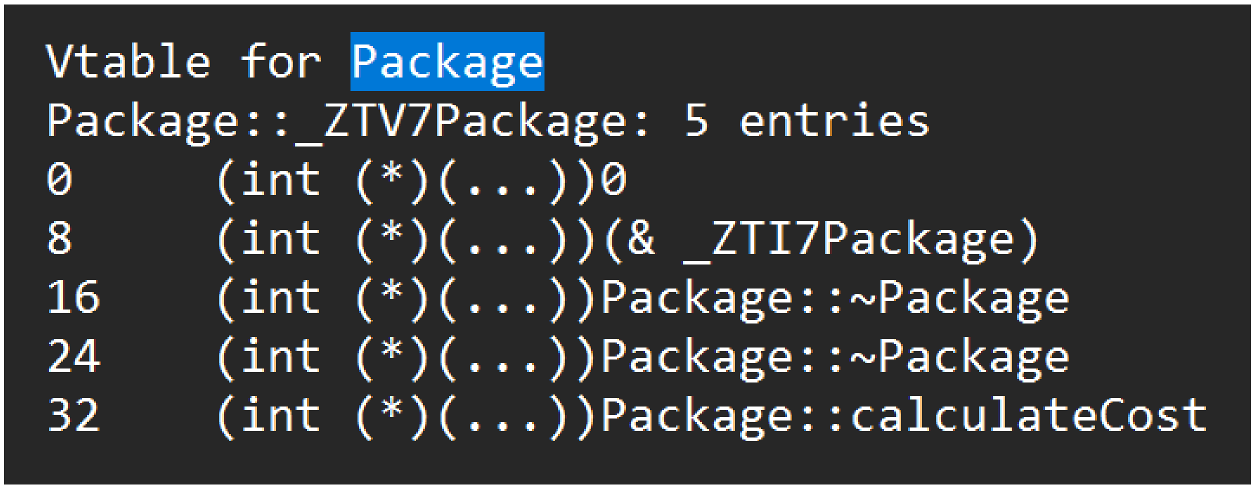
While this approach might generally work out for small scale projects, as soon as a program's structure begins to grow in complexity this "shoot from the hip" approach quickly loses its luster. The lack of a clear thought-out structure can inadvertently lead to redundant code, an issue that just get's exponentially worse when working in a team. Even the best-case scenario of a small-scale project can benefit from a design phase, this assignment is a perfect example of that.

Question 1's assignment guidelines mention that the package class needs to have elements for both the sender and receiver's address information. This would lead to a relatively large list of attributes, which can lead to hard-to-understand code. While designing the UML diagram displayed in Fig.1, I decided to diverge form the guidelines instead creating an Address class with a getInfo() function. Following this, I created a pointer to an array of 2 Address objects in the Package class, which results in much cleaner code than a long list of attributes would have.

Task 1.4

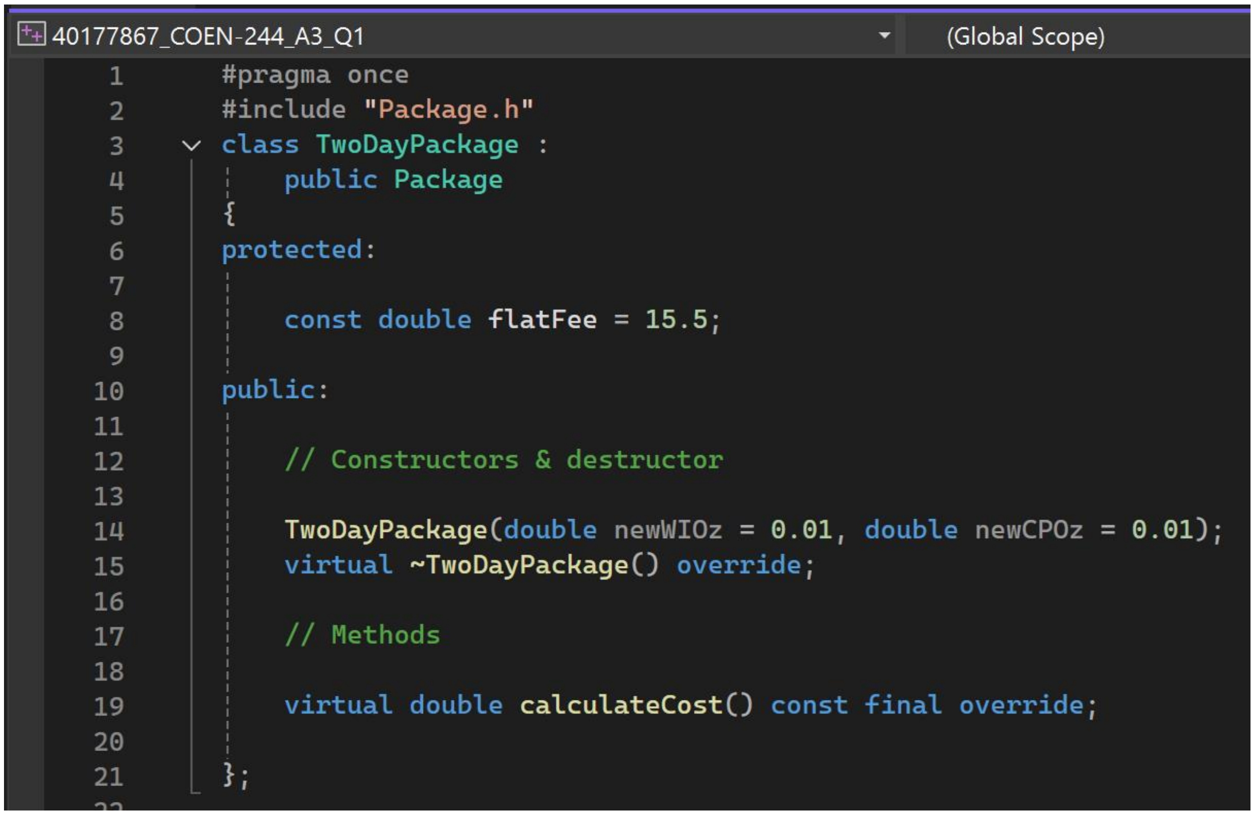


```cpp
#pragma once
#include "Address.h"
#include <array>
#include <stdexcept>
#include <iomanip>
#include <sstream>

class Package
{
protected:
    Address* toFrom;

    double weightInOz;
    double costPerOz;

public:

    // Constructors & destructor

    Package(double newWIOz = 0.01, double newCPOz = 0.01);
    virtual ~Package();

    // Setters

    void setWeightInOz(const double);
    void setCostPerOz(const double);


    // Getters

    double getWeightInOz() const;
    double getCostPerOz() const;

    // Methods

    virtual double calculateCost() const;
    void setToFrom(int, const std::string, const std::string, const std::string, const std::string, const std::string);
    std::string getToFrom(int) const;

};
```



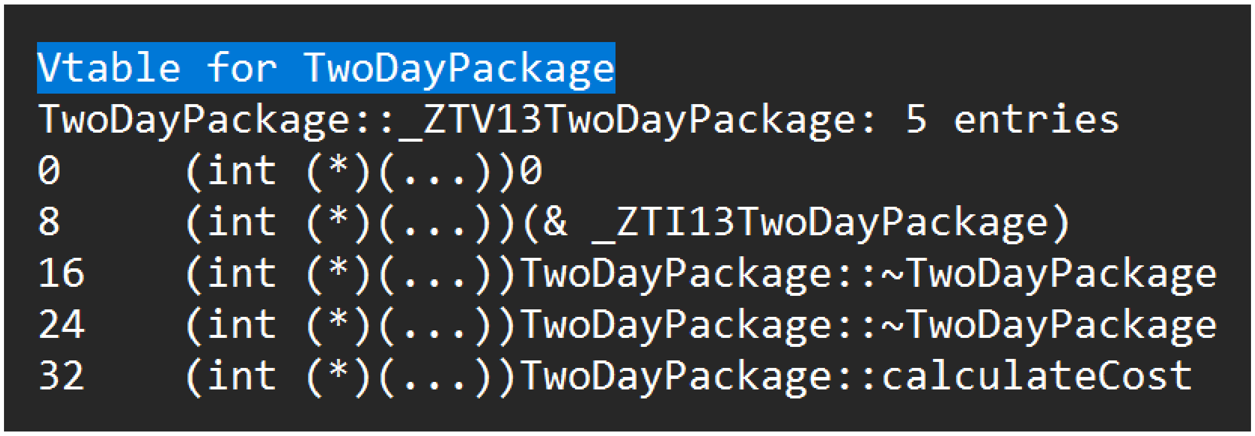
```
Vtable for Package
Package::_ZTV7Package: 5 entries
0     (int (*)(...))0
8     (int (*)(...))(& _ZTI7Package)
16    (int (*)(...))Package::~Package
24    (int (*)(...))Package::~Package
32    (int (*)(...))Package::calculateCost
```

Fig1.2: Package Class and Vtable.



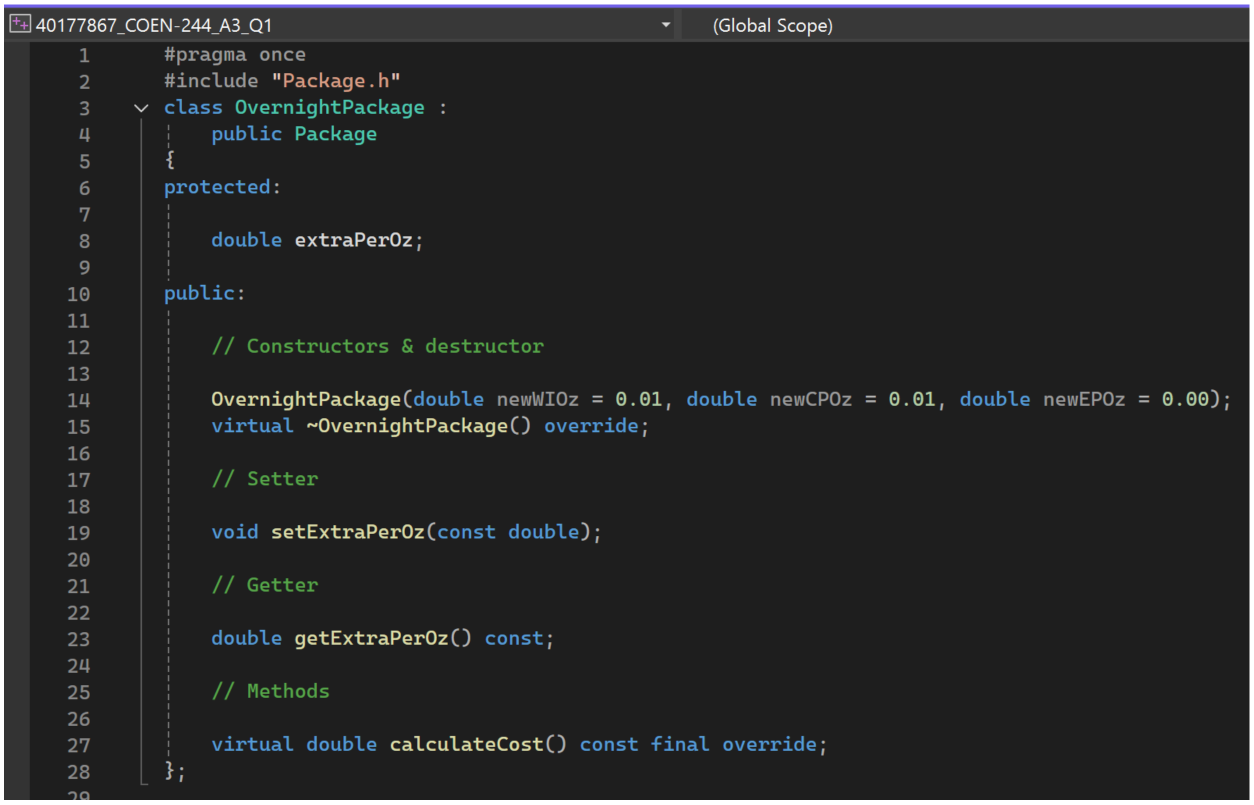
40177867_COEN-244_A3_Q1 (Global Scope)

```cpp
#pragma once
#include "Package.h"
class TwoDayPackage :
    public Package
{
protected:

    const double flatFee = 15.5;

public:

    // Constructors & destructor

    TwoDayPackage(double newWIOz = 0.01, double newCPOz = 0.01);
    virtual ~TwoDayPackage() override;

    // Methods

    virtual double calculateCost() const final override;

};
```



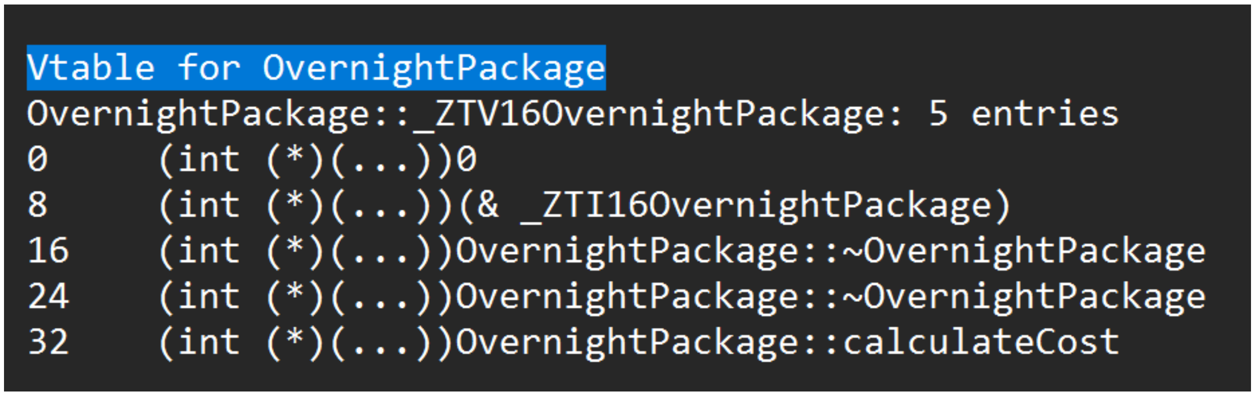
```
Vtable for TwoDayPackage
TwoDayPackage::_ZTV13TwoDayPackage: 5 entries
0     (int (*)(...))0
8     (int (*)(...))(& _ZTI13TwoDayPackage)
16    (int (*)(...))TwoDayPackage::~TwoDayPackage
24    (int (*)(...))TwoDayPackage::~TwoDayPackage
32    (int (*)(...))TwoDayPackage::calculateCost
```

Fig 1.3: TwoDayPackage Class and Vtable.



40177867_COEN-244_A3_Q1 (Global Scope)

```cpp
#pragma once
#include "Package.h"
class OvernightPackage :
    public Package
{
protected:

    double extraPerOz;

public:

    // Constructors & destructor

    OvernightPackage(double newWIOz = 0.01, double newCPOz = 0.01, double newEPOz = 0.00);
    virtual ~OvernightPackage() override;

    // Setter

    void setExtraPerOz(const double);

    // Getter

    double getExtraPerOz() const;

    // Methods

    virtual double calculateCost() const final override;
};
```



```
Vtable for OvernightPackage
OvernightPackage::_ZTV16OvernightPackage: 5 entries
0     (int (*)(...))0
8     (int (*)(...))(& _ZTI16OvernightPackage)
16    (int (*)(...))OvernightPackage::~OvernightPackage
24    (int (*)(...))OvernightPackage::~OvernightPackage
32    (int (*)(...))OvernightPackage::calculateCost
```

Fig 1.4: OvernightPackage Class and Vtable.

In the Figures 1.2-1.4 we can find screenshots of the class's header files along with their associated Vtables. We can see that the only functions are appear in the Vtables are those that were marked as virtual in the classes.