|  |
| --- |
| creeper1.jpg |
| Assignment 7 |
| CPTN230 |
|  |
| **Paul Fahey** |
| **11/27/2011** |



Contents

[Introduction 3](#_Toc310269004)

[Application Summary 3](#_Toc310269005)

[Object and Variable Functionality 4](#_Toc310269006)

[Pseudo Code 4](#_Toc310269007)

[Assignment 7 Monster Header File 4](#_Toc310269008)

[Assignment 7 Undead Header File 5](#_Toc310269009)

[Assignment 7 Zombie Header File 5](#_Toc310269010)

[Assignment 7 Skeleton Header File 5](#_Toc310269011)

[Assignment 7 Monster Class File 6](#_Toc310269012)

[Assignment 7 Undead Class File 6](#_Toc310269013)

[Assignment 7 Zombie Class File 7](#_Toc310269014)

[Assignment 7 Skeleton Class File 7](#_Toc310269015)

[Assignment 7 Application File 7](#_Toc310269016)

[Conclusions 8](#_Toc310269017)

[References 9](#_Toc310269018)

# Introduction

This assignment is about demonstrating public inheritance. A series of classes will be created to create an inheritance scheme. The classes will create different but related sets of objects in the application.

# Application Summary

This program will demonstrate inheritance by using a monster, undead, zombie, and skeleton class. The user will first be greeted to the program. Afterwards, a line of text will appear on the screen for each object representing each classes data member's value along with the values that were set for its inherited data member. Also the program will display a mini message combining the values of the data types of each of those classes. A thank message will appear at the end of the program.

Monster

Member: Darkness

Undead

Member: Alive

Zombie

Member: Flesh

Skeleton

Member: Bones

|  |  |  |
| --- | --- | --- |
| **Class** | **Object** | **Test** |
| Monster | Big | darkness = "to roam in the dark" |
| Undead | Average | darkness = " to sit in the dark |
| Undead | Average | alive = "are dead, but act alive" |
| Zombie | Walk | darkness = " to walk in the dark |
| Zombie | Walk | alive = "are dead, but live life" |
| Zombie | Walk | flesh = "walking weird, and eat flesh" |
| Skeleton | Skinny | darkness = " to run in the dark" |
| Skeleton | Skinny | alive = "definitely look dead but are well alive" |
| Skeleton | Skinny | flesh = "having no flesh for a while" |
| Skeleton | Skinny | bones = "have their bones showing" |

# Object and Variable Functionality

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Monster Object** | **Undead Object** | **Zombie Object** | **Skeleton Object** |
| **Data Member:** | Darkness | Alive | Flesh | Bones |
| **Data Type:** | String | String | String | String |
| **Initial value:** | Empty string | Empty string | Empty string | Empty string |
| **Purpose:** | Set a darkness for monster and its children | Set a state of living for undead and its children | Set flesh for zombie and its children | Set bones for a skeleton |

# Pseudo Code

## Assignment 7 Monster Header File

* A class called monster is created

**Data Members (Private):**

* A *darkness* with type string

**Member Functions (Public):**

* A default **monster** constructor that will initialize all the data members
* A monster destructor that will destroy the monster objects
* A **set** function to set the *darkness* that will not return any value but take in a string value
* A **get** function to get the *darkness* that will return an string value but not take in any value

## Assignment 7 Undead Header File

* Will include the monster class
* A class called undead is created
* The undead class will publicly inherit the monster class

**Data Members (Private):**

* A *alive* with type string

**Member Functions (Public):**

* A default **undead** constructor that will initialize all the data members
* A undead destructor that will destroy the undead objects
* A **set** function to set the *alive* that will not return any value but take in a string value
* A **get** function to get the *alive* that will return an string value but not take in any value

## Assignment 7 Zombie Header File

* Will include the undead class
* A class called zombie is created
* The zombie class will publicly inherit the undead class

**Data Members (Private):**

* A *flesh* with type string

**Member Functions (Public):**

* A default **zombie** constructor that will initialize all the data members
* A zombie destructor that will destroy the zombie objects
* A **set** function to set the *flesh* that will not return any value but take in a string value
* A **get** function to get the *flesh* that will return an string value but not take in any value

## Assignment 7 Skeleton Header File

* Will include the zombie class
* A class called skeleton is created
* The skeleton class will publicly inherit the zombie class

**Data Members (Private):**

* A *bones* with type string

**Member Functions (Public):**

* A default **skeleton** constructor that will initialize all the data members
* A skeleton destructor that will destroy the skeleton objects
* A **set** function to set the *bones* that will not return any value but take in a string value
* A **get** function to get the *bones* that will return an string value but not take in any value

## Assignment 7 Monster Class File

* Includes the monster class header file

Default **monster** constructor initializes the data members:

* A message that will explicitly display the constructors existence when called
* *darkness* is set with an string value of empty string

**Monster** destructor that will destroy the monster objects:

* A message to show the destructors existence when called

The **set darkness** function is initialized that takes in a *input* darkness as a parameter

* The *input* darkness is given as the value for data member *darkness*

The **get** **darkness** function is initialized that takes in no parameters

* It returns the *darkness* data member to the application file

## Assignment 7 Undead Class File

* Includes the undead class header file

Default **undead** constructor initializes the data members:

* A message that will explicitly display the constructors existence when called
* *alive* is set with an string value of empty string

**Undead** destructor that will destroy the undead objects:

* A message to show the destructors existence when called

The **set alive** function is initialized that takes in a *input* alive as a parameter

* The *input* alive is given as the value for data member *alive*

The **get** **alive** function is initialized that takes in no parameters

* It returns the *alive* data member to the application file

## Assignment 7 Zombie Class File

* Includes the zombie class header file

Default **zombie** constructor initializes the data members:

* A message that will explicitly display the constructors existence when called
* *flesh* is set with an string value of empty string

**Zombie** destructor that will destroy the zombie objects:

* A message to show the destructors existence when called

The **set flesh** function is initialized that takes in a *input* flesh as a parameter

* The *input* flesh is given as the value for data member *flesh*

The **get** **flesh** function is initialized that takes in no parameters

* It returns the *flesh* data member to the application file

## Assignment 7 Skeleton Class File

* Includes the skeleton class header file

Default **skeleton** constructor initializes the data members:

* A message that will explicitly display the constructors existence when called
* *bones* is set with an string value of empty string

**Skeleton** destructor that will destroy the skeleton objects:

* A message to show the destructors existence when called

The **set bones** function is initialized that takes in a *input* bones as a parameter

* The *input* bones is given as the value for data member *bones*

The **get** **bones** function is initialized that takes in no parameters

* It returns the *bones* data member to the application file

## Assignment 7 Application File

* Includes the monster class header file
* Includes the undead class header file
* Includes the zombie class header file
* Includes the skeleton class header file
* The main function begins
* A message that welcomes the user to the program
* A monster object is created
* A undead object is created
* A zombie object is created
* A skeleton object is created
* The set darkness function is called to set the darkness of the monster object
* A message is displayed that is accompanied with the get darkness function
* The set darkness function is called to set the darkness of the undead object
* The set alive function is called to set the alive of the undead object
* A message is displayed that is accompanied with the get darkness and get alive function
* The set darkness function is called to set the darkness of the zombie object
* The set alive function is called to set the alive of the zombie object
* The set flesh function is called to set the flesh of the zombie object
* A message is displayed that is accompanied with the get darkness, get alive, and get flesh function
* The set darkness function is called to set the darkness of the skeleton object
* The set alive function is called to set the alive of the skeleton object
* The set flesh function is called to set the flesh of the skeleton object
* The set bones function is called to set the bones of the skeleton object
* A message is displayed that is accompanied with the get darkness, get alive, get flesh, and get bones function
* A message is displayed thanking the user for using the program
* The main function has ended

# Conclusions

This assignment was about learning simple class inheritance. What I thought went well with this assignment was my original inheritance scheme. The scheme I chose seemed to make that proper inheritance tree so things can be inherited from its parent properly. I didn't really encounter an severe problems while doing this assignment but maybe just to be a little more cautious of the correct header file and inheritance placements since I was working with quite a few files. There was nothing that was left unresolved when building this program so I found that to be a plus. I didn't recall any parts that could have been approached with a different technique since what I did is basically in its simplest form. Finally, if I was to update this program probably the only thing would be to maybe make a more complex inheritance scheme to have the relations make a little more sense than they do now.

# References

Bettle, H. (n.d.). Assignment 7 for CPTN230 Assignment Description. 8.

Bettle, H. (n.d.). Assignment 7 Sample Files. 8.

Deitel, P., & Deitel, H. (2010). *C++ How To Program.* Upper Saddle River: Pearson Education Inc.

Fahey, P. (2011). Assignment 5 Design Document. 11.