|  |  |
| --- | --- |
| Edward Eisenberger | 1066164 |
| Zainab Al Taweel | 1062858 |
| Udaya Sree Reddy Teegulla | 1067198 |
| Assignment 4 | |
| Due Date: | 11/12/2018 |
| Date of Submission: | 11/11/2018 |

[**1. Requirements Specification**](#_x1n9t7oycrnw) **3**

[**2. Documented Source Code**](#_9sg1tsufjryr) **3**

[2.1 Contact.h](#_th14rjos7alg) 3

[2.2 Contact.cpp](#_kzewy71nwxw) 3

[2.3 PersonContact.h](#_bchquizi7plp) 4

[2.4 PersonContact.cpp](#_2lom1qon30n2) 5

[2.5 BusinessContact.h](#_bljbuaqj9e3i) 6

[2.6 BusinessContact.cpp](#_5s8zt4ygcxza) 7

[2.7 PersonAddressContact.h](#_18a4e0upypi2) 8

[2.8 PersonAddressContact.cpp](#_ghhf9eccrzto) 8

[2.9 PersonEmailContact.h](#_xzupyid5gd1l) 9

[2.10 PersonEmailContact.cpp](#_femlcgkodm30) 10

[2.11 PersonPhoneContact.h](#_fxska1rzcfic) 11

[2.12 PersonPhoneContact.cpp](#_bivmwcr0amgr) 12

[2.13 BusinessAddressContact.h](#_c7o6mn2jsoax) 13

[2.14 BusinessAddressContact.cpp](#_15y5ve7bhjmp) 14

[2.15 BusinessPhoneContact.h](#_sutobdnka4mg) 15

[2.16 BusinessPhoneContact.cpp](#_sp8u2psxv8z) 16

[2.17 BusinessWebContact.h](#_ibost5kf5sn) 17

[2.18 BusinessWebContact.cpp](#_jwog0zlarwxh) 17

[**3. Input Files**](#_8dfik991yd8d) **19**

[**4. Output Files**](#_mc8ylx3vgcqu) **19**

[**5. Screenshots**](#_lnxgbopnmk0) **19**

[**6. Lessons Learned**](#_thncceiau2o3) **19**

[**7. Work Cited**](#_b80932rwyl9) **19**

# 1. Requirements Specification

1. Implement Contact Hierarchy
2. Implement display method which indicates prints the object’s type

# 2. Documented Source Code

## 2.1 Contact.h

/\*\*

\* @file Contact.h

\* @author Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Contact class interface

\*/

#ifndef Contact\_h

#define Contact\_h

#include <string>

class Contact

{

private:

std::string voName;

public:

Contact( const std::string& aorName = "" );

Contact( const Contact& aorContact );

virtual ~Contact( void );

Contact& operator=( const Contact& aorContact );

virtual void display( void );

};

#endif

## 2.2 Contact.cpp

/\*\*

\* @file Contact.cpp

\* @author Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Contact class implementation

\*/

#include "Contact.h"

#include <string>

#include <iostream>

Contact::Contact( const std::string& aorName )

{

this->voName = aorName;

}

Contact::Contact( const Contact& aorContact )

{

// Call assignment operator

\*this = aorContact;

}

Contact::~Contact( void )

{

// Nothing to destruct

}

Contact& Contact::operator=( const Contact& aorContact )

{

// Copy Contact attributes

this->voName = aorContact.voName;

return( \*this );

}

void Contact::display( void )

{

std::cout << "Contact" << std::endl;

}

## 2.3 PersonContact.h

/\*\*

\* @file PersonContact.h

\* @author Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Person Contact class interface

\*/

#ifndef PersonContact\_h

#define PersonContact\_h

#include "Contact.h"

#include <string>

class PersonContact : public Contact

{

private:

std::string voGender;

public:

PersonContact( const std::string& aorName = "", const std::string& aorGender = "" );

PersonContact( const PersonContact& aorContact );

virtual ~PersonContact( void );

PersonContact& operator=( const PersonContact& aorPerson );

virtual void display( void );

};

#endif

## 2.4 PersonContact.cpp

/\*\*

\* @file PersonContact.cpp

\* @author Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Person Contact class implementation

\*/

#include "PersonContact.h"

#include <string>

#include <iostream>

PersonContact::PersonContact( const std::string& aorName, const std::string& aorGender )

: Contact( aorName )

{

this->voGender = aorGender;

}

PersonContact::PersonContact( const PersonContact& aorContact )

{

// Call assignment operator

\*this = aorContact;

}

PersonContact::~PersonContact( void )

{

// Nothing to destruct

}

PersonContact& PersonContact::operator=( const PersonContact& aorPerson )

{

// Copy Contact attributes

Contact::operator=( static\_cast< Contact >( aorPerson ) );

// Copy PersonContact attributes

this->voGender = aorPerson.voGender;

return( \*this );

}

void PersonContact::display( void )

{

std::cout << "Person Contact \t -> ";

Contact::display( );

}

## 2.5 BusinessContact.h

/\*\*

\* @file BusinessContact.h

\* @author Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Business Contact class interface

\*/

#ifndef BusinessContact\_h

#define BusinessContact\_h

#include "Contact.h"

class BusinessContact : public Contact

{

private:

std::string voType;

public:

BusinessContact( const std::string& aorName = "", const std::string& aorType = "" );

BusinessContact( const BusinessContact& aorBusiness );

virtual ~BusinessContact( void );

BusinessContact& operator=( const BusinessContact& aorBusiness );

virtual void display( void );

};

#endif

## 2.6 BusinessContact.cpp

/\*\*

\* @file BusinessContact.cpp

\* @author Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Business Contact class implementation

\*/

#include "BusinessContact.h"

#include <string>

#include <iostream>

BusinessContact::BusinessContact( const std::string& aorName, const std::string& aorType )

: Contact( aorName )

{

this->voType = aorType;

}

BusinessContact::BusinessContact( const BusinessContact& aorBusiness )

{

// Call assignment operator

\*this = aorBusiness;

}

BusinessContact::~BusinessContact( void )

{

// Nothing to destruct

}

BusinessContact& BusinessContact::operator=( const BusinessContact& aorBusiness )

{

// Copy Contact attributes

Contact::operator=( static\_cast< Contact >( aorBusiness ) );

// Copy Business Contact attributes

this->voType = aorBusiness.voType;

return( \*this );

}

void BusinessContact::display( void )

{

std::cout << "Business Contact \t -> ";

Contact::display( );

}

## 2.7 PersonAddressContact.h

/\*\*

\* @file PersonAddressContact.h

\* @author Zainab Al Taweel, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Person Address Contact class interface

\*/

#ifndef PersonAddressContact\_h

#define PersonAddressContact\_h

#include <string>

#include <iostream>

#include "PersonContact.h"

using namespace std;

class PersonAddressContact : public PersonContact

{

public:

PersonAddressContact(const string & oName="", const string & oGender = "", const string & oAddress = ""); // Default constructor

PersonAddressContact(const PersonAddressContact & oPAC); // Copy Constructor

PersonAddressContact& operator = (const PersonAddressContact & oPAC); // = operator overloading

virtual void display(void); // modified display functoin

virtual ~PersonAddressContact(); // Virtual Destructor

private:

string Address;

};

#endif

## 2.8 PersonAddressContact.cpp

/\*\*

\* @file PersonAddressContact.cpp

\* @author Zainab Al Taweel, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Person Address Contact class implementation

\*/

#include "PersonAddressContact.h"

#include <string>

#include <iostream>

using namespace std;

PersonAddressContact::PersonAddressContact(const string & oName, const string & oGender, const string & oAddress)

: PersonContact(oName, oGender) // Deafult Constructor

{

this->Address = oAddress;

}

PersonAddressContact::PersonAddressContact(const PersonAddressContact & oPAC) // Copy constructor

{

\*this = oPAC;

}

PersonAddressContact & PersonAddressContact::operator=(const PersonAddressContact & oPAC) // Copy Constructor

{

PersonContact::operator=(static\_cast<PersonContact>(oPAC));

this->Address = oPAC.Address;

return \*this;

}

void PersonAddressContact::display(void)

{

cout << "Person Address Contact \t -> ";

PersonContact::display( );

}

PersonAddressContact::~PersonAddressContact() // Destructor

{

}

## 2.9 PersonEmailContact.h

/\*\*

\* @file PersonEmailContact.h

\* @author Zainab Al Taweel, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Person Email Contact class interface

\*/

#ifndef PersonEmailContact\_h

#define PersonEmailContact\_h

#include <string>

#include <iostream>

#include "PersonContact.h"

using namespace std;

class PersonEmailContact : public PersonContact

{

public:

PersonEmailContact(const string & oName = "", const string & oGender = "", const string & oEmail = ""); // Default constructor

PersonEmailContact(const PersonEmailContact & oPEC); // Copy Constructor

PersonEmailContact& operator = (const PersonEmailContact & oPEC); // = operator overloading

virtual void display(void); // modified display functoin

virtual ~PersonEmailContact(); // Virtual Destructor

private:

string Email;

};

#endif

## 2.10 PersonEmailContact.cpp

/\*\*

\* @file PersonEmailContact.cpp

\* @author Zainab Al Taweel, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Person Email Contact class implementation

\*/

#include "PersonEmailContact.h"

#include <string>

#include <iostream>

using namespace std;

PersonEmailContact::PersonEmailContact(const string & oName, const string & oGender, const string & oEmail)

: PersonContact(oName, oGender) // Deafult Constructor

{

this->Email = oEmail;

}

PersonEmailContact::PersonEmailContact(const PersonEmailContact & oPEC) // Copy constructor

{

\*this = oPEC;

}

PersonEmailContact & PersonEmailContact::operator=(const PersonEmailContact & oPEC) // Copy Constructor

{

PersonContact::operator=(static\_cast<PersonContact>(oPEC));

this->Email = oPEC.Email;

return \*this;

}

void PersonEmailContact::display(void)

{

cout << "Person Email Contact \t -> ";

PersonContact::display( );

}

PersonEmailContact::~PersonEmailContact() // Destructor

{

}

## 2.11 PersonPhoneContact.h

/\*\*

\* @file PersonPhoneContact.h

\* @author Zainab Al Taweel, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Person Phone Contact class interface

\*/

#ifndef PersonPhoneContact\_h

#define PersonPhoneContact\_h

#include <string>

#include <iostream>

#include "PersonContact.h"

using namespace std;

class PersonPhoneContact : public PersonContact

{

public:

PersonPhoneContact(const string & oName = "", const string & oGender = "", const int & oPhoneNo = 0); // Default constructor

PersonPhoneContact(const PersonPhoneContact & oPPC); // Copy Constructor

PersonPhoneContact& operator = (const PersonPhoneContact & oPAC); // = operator overloading

virtual void display(void); // modified display functoin

virtual ~PersonPhoneContact(); // Virtual Destructor

private:

int Phone;

};

#endif

## 2.12 PersonPhoneContact.cpp

/\*\*

\* @file PersonPhoneContact.cpp

\* @author Zainab Al Taweel, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Person Phone Contact class implementation

\*/

#include "PersonPhoneContact.h"

#include <string>

#include <iostream>

using namespace std;

PersonPhoneContact::PersonPhoneContact(const string & oName, const string & oGender, const int & oPhoneNo)

: PersonContact(oName, oGender) // Deafult Constructor

{

this->Phone = oPhoneNo;

}

PersonPhoneContact::PersonPhoneContact(const PersonPhoneContact & oPPC) // Copy constructor

{

\*this = oPPC;

}

PersonPhoneContact & PersonPhoneContact::operator=(const PersonPhoneContact & oPPC) // Copy Constructor

{

PersonContact::operator=(static\_cast<PersonContact>(oPPC));

this->Phone = oPPC.Phone;

return \*this;

}

void PersonPhoneContact::display(void)

{

cout << "Person Phone Contact \t -> ";

PersonContact::display( );

}

PersonPhoneContact::~PersonPhoneContact() // Destructor

{

}

## 2.13 BusinessAddressContact.h

/\*\*

\* @file BusinessAddressContact.h

\* @author Udaya Sree Reddy Teegulla, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Business Address Contact class interface

\*/

#ifndef BusinessAddressContact\_h

#define BusinessAddressContact\_h

#include <string>

#include <iostream>

#include "BusinessContact.h"

using namespace std;

class BusinessAddressContact : public BusinessContact

{

public:

BusinessAddressContact(const string & oName = "", const string & oGender = "", const string & oAddress = ""); // Default constructor

BusinessAddressContact(const BusinessAddressContact & oPAC); // Copy Constructor

BusinessAddressContact& operator = (const BusinessAddressContact & oPAC); // = operator overloading

virtual void display(void); // modified display functoin

virtual ~BusinessAddressContact(); // Virtual Destructor

private:

string Address;

};

#endif

## 2.14 BusinessAddressContact.cpp

/\*\*

\* @file BusinessAddressContact.cpp

\* @author Udaya Sree Reddy Teegulla, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Business Address Contact class implementation

\*/

#include "BusinessAddressContact.h"

#include <string>

#include <iostream>

using namespace std;

BusinessAddressContact::BusinessAddressContact(const string & oName, const string & oGender, const string & oAddress)

: BusinessContact(oName, oGender) // Deafult Constructor

{

this->Address = oAddress;

}

BusinessAddressContact::BusinessAddressContact(const BusinessAddressContact & oBAC) // Copy constructor

{

\*this = oBAC;

}

BusinessAddressContact & BusinessAddressContact::operator=(const BusinessAddressContact & oPAC) // Copy Constructor

{

BusinessContact::operator=(static\_cast<BusinessContact>(oPAC));

this->Address = oPAC.Address;

return \*this;

}

void BusinessAddressContact::display(void)

{

cout << "Business Address Contact \t -> ";

BusinessContact::display( );

}

BusinessAddressContact::~BusinessAddressContact() // Destructor

{

}

## 2.15 BusinessPhoneContact.h

/\*\*

\* @file BusinessPhoneContact.h

\* @author Udaya Sree Reddy Teegulla, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Business Phone Contact class interface

\*/

#ifndef BusinessPhoneContact\_h

#define BusinessPhoneContact\_h

#include <string>

#include <iostream>

#include "BusinessContact.h"

using namespace std;

class BusinessPhoneContact : public BusinessContact

{

public:

BusinessPhoneContact(const string & oName = "", const string & oGender = "", const int & oPhoneNo = 0); // Default constructor

BusinessPhoneContact(const BusinessPhoneContact & oPPC); // Copy Constructor

BusinessPhoneContact& operator = (const BusinessPhoneContact & oPAC); // = operator overloading

virtual void display(void); // modified display functoin

virtual ~BusinessPhoneContact(); // Virtual Destructor

private:

int Phone;

};

#endif

## 2.16 BusinessPhoneContact.cpp

/\*\*

\* @file BusinessPhoneContact.cpp

\* @author Udaya Sree Reddy Teegulla, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Business Phone Contact class implementation

\*/

#include "BusinessPhoneContact.h"

#include <string>

#include <iostream>

using namespace std;

BusinessPhoneContact::BusinessPhoneContact(const string & oName, const string & oGender, const int & oPhoneNo)

: BusinessContact(oName, oGender) // Deafult Constructor

{

this->Phone = oPhoneNo;

}

BusinessPhoneContact::BusinessPhoneContact(const BusinessPhoneContact & oBPC) // Copy constructor

{

\*this = oBPC;

}

BusinessPhoneContact & BusinessPhoneContact::operator=(const BusinessPhoneContact & oBPC) // Copy Constructor

{

BusinessContact::operator=(static\_cast<BusinessContact>(oBPC));

this->Phone = oBPC.Phone;

return \*this;

}

void BusinessPhoneContact::display(void)

{

cout << "Business Phone Contact \t -> ";

BusinessContact::display( );

}

BusinessPhoneContact::~BusinessPhoneContact() // Destructor

{

}

## 2.17 BusinessWebContact.h

/\*\*

\* @file BusinessWebContact.h

\* @author Udaya Sree Reddy Teegulla, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Business Web Contact class interface

\*/

#ifndef BusinessWebContact\_h

#define BusinessWebContact\_h

#include <string>

#include <iostream>

#include "BusinessContact.h"

using namespace std;

class BusinessWebContact : public BusinessContact

{

public:

BusinessWebContact(const string & oName = "", const string & oGender = "", const string & oWebsite = ""); // Default constructor

BusinessWebContact(const BusinessWebContact & oBWC); // Copy Constructor

BusinessWebContact& operator = (const BusinessWebContact & oBWC); // = operator overloading

virtual void display(void); // modified display functoin

virtual ~BusinessWebContact(); // Virtual Destructor

private:

string Website;

};

#endif

## 2.18 BusinessWebContact.cpp

/\*\*

\* @file BusinessWebContact.cpp

\* @author Udaya Sree Reddy Teegulla, Edward Eisenberger

\* @date 2018-11-11

\* @compiler Visual C++ 2017

\*

\* @brief Business Web Contact class implementation

\*/

#include "BusinessWebContact.h"

#include <string>

#include <iostream>

using namespace std;

BusinessWebContact::BusinessWebContact(const string & oName, const string & oGender, const string & oWebsite)

: BusinessContact(oName, oGender) // Deafult Constructor

{

this->Website = oWebsite;

}

BusinessWebContact::BusinessWebContact(const BusinessWebContact & oBWC) // Copy constructor

{

\*this = oBWC;

}

BusinessWebContact & BusinessWebContact::operator=(const BusinessWebContact & oBWC) // Copy Constructor

{

BusinessContact::operator=(static\_cast<BusinessContact>(oBWC));

this->Website = oBWC.Website;

return \*this;

}

void BusinessWebContact::display(void)

{

cout << "Business Web Contact \t -> ";

BusinessContact::display( );

}

BusinessWebContact::~BusinessWebContact() // Destructor

{

}

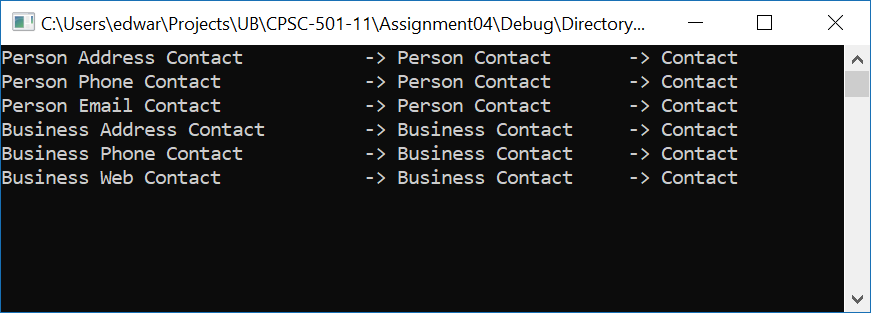
# 3. Input Files

No input files were used.

# 4. Output Files

No output files were written to.

# 5. Screenshots



# 6. Lessons Learned

1. Class hierarchy
2. Inheritance
3. Virtual methods
4. Polymorphism

# 7. Work Cited

None