Program 1: Inheritance (30 Points)

**Objectives:**

The purpose of this assignment is to better understand the concepts and reasoning for Inheritance

**Program Background:**

Complex entities often exist in a **hierarchy** with common similarities. When designing entities for things such as new user defined data types the common foundations between these entities can be taken advantage of. In our case, the concept of Inheritance.

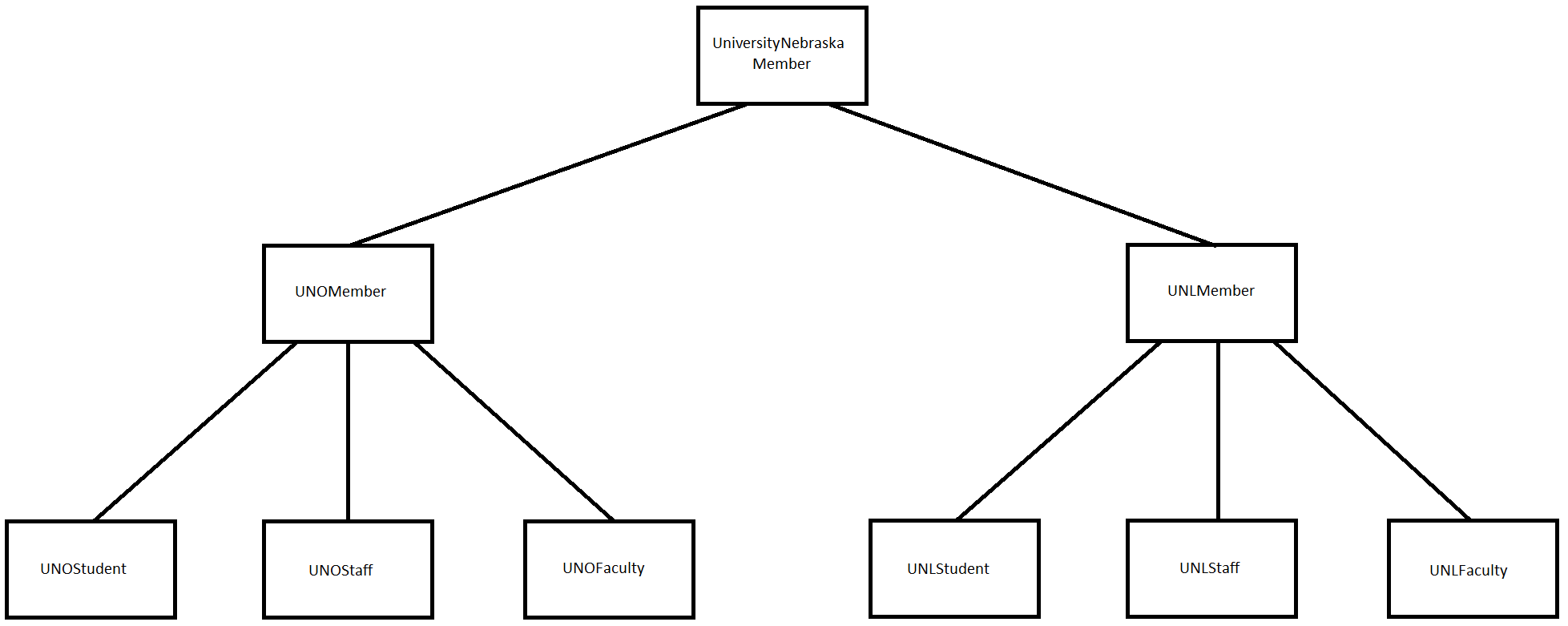
**Program Description:**

For this assignment you will create a simple class hierarchy to represent types of people that are in the Nebraska University system. The focus will be in the design and creating the types themselves, not creating a “useful” program. A test driver file is provided to ensure the classes have been created correctly.

The following classes are required:

* UniversityNebraskaMember
* UNOMember
* UNLMember
* UNOStudent
* UNOStaff
* UNOFaculty
* UNLStudent
* UNLStaff
* UNLFaculty
* UniversityNebraskaDriver (provided)

The hierarchy will look like this (full UML descriptions below):



**UML DIAGRAM FOR AND DISCUSSION FOR UniversityNebraskaMember**

|  |
| --- |
| UniversityNebraskaMember |
| * name : String * NUID : int * city : String |
| <<constructor>> UniversityNebraskaMember( name : String, NUID : int, city : String )  + setName( name : String )  + setCity( city : String )  + setNUID( NUID : int )  + toString( ) : String |

UniversityNebraskaMember is the **super class** to all others. It contains those data members and behaviors that are common to all of those that are in the University of Nebraska system.

Data Members:

name – Person’s name

NUID – Unique numerical identifier across the entire University of Nebraska system

city – The city of the person’s campus

Methods:

UniversityNebraskMember(name : String, NUID : int, city : String) – Class constructor. Accepts arguments for all three data members. Calls upon the below set methods.

setName( name : String ) – Sets the name data member to the passed value

setCity( name : String ) – Sets the city data member to the passed value

setNUID( NUID : int ) – An NUID must be an 8 digit number. If the value passed is an 8 digit number it will set the NUID data member to that value, otherwise it will set the data member to -1 to denote an invalid NUID

toString( ) – Returns a String of the following format (uppercase denotes data member values to be output):

Name: NAME

NUID: NUID

Campus Location: CITY

If the value for NUID is invalid (-1) instead put “NOT SET” for the NUID:

Name: NAME

NUID: NOT SET

Campus Location: CITY

UniversityNebraskaMember has two direct subclasses, one for UNO members and one for UNL members. The difference between the two is the type of ID used for those specific campuses.

**UML DIAGRAM FOR AND DISCUSSION FOR UNOMember**

|  |
| --- |
| UNOMember **extends** UniversityNebraskaMember |
| * netID : String |
| <<constructor>> UNOMember( name : String, NUID : int, netID : String )  + setNetID( netID : String )  + toString( ) : String |

Data Members:

netID – netID is the unique identifier used specifically within the UNO system

Methods:

UNOMember(name : String, NUID : int, netID : String ) – Class constructor. Calls upon **super class constructor** with the given name, NUID, and city as Omaha. Sets netID using set method.

setNetID( netID : String ) – Sets the netID data member to the value passed. If **null** is passed set netID to the String “NOT SET”

toString( ) **–** Uses **super class toString()** to return a String of the following format (uppercase denotes data member values to be output):

Name: NAME

NUID: NUID

Campus Location: CITY

NetID: NETID

**UML DIAGRAM FOR AND DISCUSSION FOR UNLMember**

|  |
| --- |
| UNLMember **extends** UniversityNebraskaMember |
| * myUNLID : String |
| <<constructor>> UNLMember( name : String, NUID : int, myUNLID : String )  + setMyUNLID( myUNLID : String )  + toString( ) : String |

Data Members:

myUNLID – myUNLID is the unique identifier used specifically within the UNL system

Methods:

UNLMember(name : String, NUID : int, myUNLID : String ) – Class constructor. Calls upon **super class constructor** with the given name, NUID, and city as Lincoln. Sets myUNLID using set method.

setMyUNLID( myUNLID : String ) – Sets the myUNLID data member to the value passed. If **null** is passed set myUNLID to the String “NOT SET”

toString( ) **–** Uses **super class toString()** to return a String of the following format (uppercase denotes data member values to be output):

Name: NAME

NUID: NUID

Campus Location: CITY

MyUNLID: MYUNLID

Within the UNO and UNO sub-systems there are three possible types, student, staff, and faculty. Each **inherit** everything from their respective UNO/UNLMember class. These **subclasses** differentiate the specifics between types of people that are in the system.

**UML DIAGRAM FOR AND DISCUSSION FOR UNOStudent**

|  |
| --- |
| UNOStudent **extends** UNOMember |
| * balance : double * credits : int * GPA : double |
| <<constructor>> UNOStudent( name : String, NUID : int, netID : String, balance : double, credits : int, GPA : double )  + setBalance( balance : double )  + setCredits( credits : int )  + setGPA( GPA : double )  + toString( ) : String |

Data Members:

balance – Balance the student owes

credits – Total credits the student has earned

GPA – Current GPA of the student

Methods:

UNOStudent(name : String, NUID : int, netID : String, balance : double, credits : int, GPA : double) – Class constructor. Takes everything that is required to describe a UNO student. Calls upon **super class constructor** passing name, NUID, and netID. Uses set methods for local data members

setBalance( balance : double ) – Sets the balance data member to the value passed. If the passed value is less than 0, sets balance to 0

setCredits( credits : int ) – Sets the credits data member to the value passed. If the passed value is less than 0, sets credits to 0

setGPA( GPA : double ) – Sets the GPA data member to the value passed. If the passed value is not between the range of 0.0 and 4.0, sets GPA to 0

toString( ) **–** Uses **super class toString()** to return a String of the following format (uppercase denotes data member values to be output). Note the $ for balance, this value must also be formatted to 2 decimal places as is expected for monetary values:

Name: NAME

NUID: NUID

Campus Location: CITY

NetID: NETID

Balance: $BALANCE

Credits: CREDITS

GPA: GPA

**UML DIAGRAM FOR AND DISCUSSION FOR UNOStaff**

|  |
| --- |
| UNOStaff **extends** UNOMember |
| * postion : String * wage : double |
| <<constructor>> UNOStaff( name : String, NUID : int, netID : String, position : String, wage : double)  + setPosition( position : String )  + setWage( wage : double )  + toString( ) : String |

Data Members:

postion – The name of the position the Staff member holds

wage – Hourly wage of the Staff member

Methods:

UNOStaff(name : String, NUID : int, netID : String, position : String, wage : double ) – Class constructor. Takes everything that is required to describe a UNO staff member. Calls upon **super class constructor** passing name, NUID, and netID. Uses set methods for local data members

setPosition( position : String ) - Sets the position data member to the value passed. If **null** is passed set position to the String “NOT SET”

setWage( wage : double ) – Sets the wage data member to the value passed. The minimum wage is $9.00, if the passed value is less than this set wage data member to -1 to denote an invalid value

toString( ) **–** Uses **super class toString()** to return a String of the following format (uppercase denotes data member values to be output). Note the $ for wage, this value must also be formatted to 2 decimal places as is expected for monetary values:

Name: NAME

NUID: NUID

Campus Location: CITY

NetID: NETID

Position: POSITION

Wage: $WAGE

If the value for wage is invalid (-1) instead put “NOT SET” for the wage (no $):

Name: NAME

NUID: NUID

Campus Location: CITY

NetID: NETID

Position: POSITION

Wage: NOT SET

**UML DIAGRAM FOR AND DISCUSSION FOR UNOFaculty**

|  |
| --- |
| UNOFaculty **extends** UNOMember |
| * department : String * title : String * salary : double |
| <<constructor>> UNOFaculty( name : String, NUID : int, netID : String, department : String, title : String, salary : double )  + setDepartment( department : String )  + setTitle( title : String )  + setSalary( salary : double )  + toString( ) : String |

Data Members:

department – The department within the University the Faculty is in

title – The Faculty member’s title

salary – The Faculty member’s yearly salary

Methods:

UNOFaculty(name : String, NUID : int, netID : String, department : String, title : String, salary : double ) – Class constructor. Takes everything that is required to describe a UNO faculty member. Calls upon **super class constructor** passing name, NUID, and netID. Uses set methods for local data members

setDepartment( department : String ) – Sets the department data member to the value passed. If **null** is passed set department to the String “NOT SET”

setTitle( title : String ) – Sets the title data member to the value passed. If **null** is passed set title to the String “NOT SET”

setSalary( salary : double ) – Sets the salary data member to the value passed. The minimum salary is $23,660, if the passed value is less than this set salary data member to -1 to denote an invalid value

toString( ) **–** Uses **super class toString()** to return a String of the following format (uppercase denotes data member values to be output). Note the $ for salary, this value must also be formatted to 2 decimal places as is expected for monetary values:

Name: NAME

NUID: NUID

Campus Location: CITY

NetID: NETID

Department: DEPARTMENT

Title: TITLE

Salary: $SALARY

If the value for salary is invalid (-1) instead put “NOT SET” for the salary (no $):

Name: NAME

NUID: NUID

Campus Location: CITY

NetID: NETID

Department: DEPARTMENT

Title: TITLE

Salary: NOT SET

**UML DIAGRAM FOR AND DISCUSSION FOR UNLStudent**

|  |
| --- |
| UNLStudent **extends** UNLMember |
| * balance : double * credits : int * GPA : double |
| <<constructor>> UNLStudent( name : String, NUID : int, myUNLID : String, balance : double, credits : int, GPA : double )  + setBalance( balance : double )  + setCredits( credits : int )  + setGPA( GPA : double )  + toString( ) : String |

Data Members:

balance – Balance the student owes

credits – Total credits the student has earned

GPA – Current GPA of the student

Methods:

UNLStudent(name : String, NUID : int, myUNLID : String, balance : double, credits : int, GPA : double) – Class constructor. Takes everything that is required to describe a UNL student. Calls upon **super class constructor** passing name, NUID, and myUNLID. Uses set methods for local data members

setBalance( balance : double ) – Sets the balance data member to the value passed. If the passed value is less than 0, sets balance to 0

setCredits( credits : int ) – Sets the credits data member to the value passed. If the passed value is less than 0, sets credits to 0

setGPA( GPA : double ) – Sets the GPA data member to the value passed. If the passed value is not between the range of 0.0 and 4.0, sets GPA to 0

toString( ) **–** Uses **super class toString()** to return a String of the following format (uppercase denotes data member values to be output). Note the $ for balance, this value must also be formatted to 2 decimal places as is expected for monetary values:

Name: NAME

NUID: NUID

Campus Location: CITY

MyUNLID: MYUNLID

Balance: $BALANCE

Credits: CREDITS

GPA: GPA

**UML DIAGRAM FOR AND DISCUSSION FOR UNLStaff**

|  |
| --- |
| UNLStaff **extends** UNLMember |
| * postion : String * wage : double |
| <<constructor>> UNLStaff( name : String, NUID : int, myUNLID : String, position : String, wage : double)  + setPosition( position : String )  + setWage( wage : double )  + toString( ) : String |

Data Members:

postion – The name of the position the Staff member holds

wage – Hourly wage of the Staff member

Methods:

UNLStaff(name : String, NUID : int, myUNLID : String, position : String, wage : double ) – Class constructor. Takes everything that is required to describe a UNL staff member. Calls upon **super class constructor** passing name, NUID, and myUNLID. Uses set methods for local data members

setPosition( position : String ) - Sets the position data member to the value passed. If **null** is passed set position to the String “NOT SET”

setWage( wage : double ) – Sets the wage data member to the value passed. The minimum wage is $9.00, if the passed value is less than this set wage data member to -1 to denote an invalid value

toString( ) **–** Uses **super class toString()** to return a String of the following format (uppercase denotes data member values to be output). Note the $ for wage, this value must also be formatted to 2 decimal places as is expected for monetary values:

Name: NAME

NUID: NUID

Campus Location: CITY

MyUNLID: MYUNLID

Position: POSITION

Wage: $WAGE

If the value for wage is invalid (-1) instead put “NOT SET” for the wage (no $):

Name: NAME

NUID: NUID

Campus Location: CITY

MyUNLID: MYUNLID

Position: POSITION

Wage: NOT SET

**UML DIAGRAM FOR AND DISCUSSION FOR UNLFaculty**

|  |
| --- |
| UNLFaculty **extends** UNLMember |
| * department : String * title : String * salary : double |
| <<constructor>> UNLFaculty( name : String, NUID : int, myUNLID : String, department : String, title : String, salary : double )  + setDepartment( department : String )  + setTitle( title : String )  + setSalary( salary : double )  + toString( ) : String |

Data Members:

department – The department within the University the Faculty is in

title – The Faculty member’s title

salary – The Faculty member’s yearly salary

Methods:

UNLFaculty(name : String, NUID : int, myUNLID : String, department : String, title : String, salary : double ) – Class constructor. Takes everything that is required to describe a UNL faculty member. Calls upon **super class constructor** passing name, NUID, and myUNLID. Uses set methods for local data members

setDepartment( department : String ) – Sets the department data member to the value passed. If **null** is passed set department to the String “NOT SET”

setTitle( title : String ) – Sets the title data member to the value passed. If **null** is passed set title to the String “NOT SET”

setSalary( salary : double ) – Sets the salary data member to the value passed. The minimum salary is $23,660, if the passed value is less than this set salary data member to -1 to denote an invalid value

toString( ) **–** Uses **super class toString()** to return a String of the following format (uppercase denotes data member values to be output). Note the $ for salary, this value must also be formatted to 2 decimal places as is expected for monetary values:

Name: NAME

NUID: NUID

Campus Location: CITY

MyUNLID: MYUNLID

Department: DEPARTMENT

Title: TITLE

Salary: $SALARY

If the value for salary is invalid (-1) instead put “NOT SET” for the salary (no $):

Name: NAME

NUID: NUID

Campus Location: CITY

MyUNLID: MYUNLID

Department: DEPARTMENT

Title: TITLE

Salary: NOT SET

**UniversityNebraskaDriver and Other Notes**

The UniversityNebraskaDriver is provided for you. This will create instances of the **subclasses** in the hierarchy that you have created. For each instance it will check to see if what is produced by your toString methods matches what is expected. If it is correct it will output that the current check was OK, otherwise it will inform you that it was incorrect and provide the expected output. There is no input to the test driver so the correct output will always be exactly as shown below. This should make it very easy to test your classes, if any checks come up as INCORRECT points will be deducted. Do not make any changes to the driver.

Remember to use the **this** reference in order to resolve ambiguity of variable names within methods. Using **this.dataMember** is how you can explicitly reference a class data member if there is a local variable with the same name.

One of the purposes of Inheritance is to avoid repetition of code. As you can see in this case, there is still some repetition between the bottom most subclasses (UNOStudent and UNLStudent have some identical methods). This would be mitigated with using **multiple inheritance**, for example if there were just a Student class such that UNOStudent could inherit from UNOMember and Student equally, and UNLStudent could inherit from UNLMember and Student equally; however, java does not allow for multiple inheritance so there will be times when some duplication is necessary.

**Package Structure**

Programs with multiple classes benefit from utilizing a package structure for organization and potential reusability.

All of your source code (your .java files) should be in their package directory. Only those .java files that do not declare a package should be in your source directory.

UniversityNebraskaMember should be declared to belong to the NUMembers package

UNOMember should be declared to belong to the NUMembers.UNO package

UNLMember should be declared to belong to the NUMembers.UNL package

UNOStudent, UNOStaff, and UNOFaculty should be declared to belong to the NUMembers.UNO.UNOTypes package

UNLStudent, UNLStaff, and UNLFaculty should be declared to belong to the NUMembers.UNL.UNLTypes package

Since these classes will belong to packages and not in the submission directory, they will need to be imported in other classes that use them.

The UniversityNebraskaMemberDriver class is not declared to be in any package and will therefore be directly accessible in the submission directory.

If you do not implement this package structure the driver will not work and the majority of points will be lost.

**Example Output (your output should match this exactly):**

Name: Kitty Pryde

NUID: 12345678

Campus Location: Omaha

NetID: kpryde

Balance: $23500.00

Credits: 45

GPA: 3.40

UNO STUDENT 1 TEST OK

Name: Piotr Rasputin

NUID: NOT SET

Campus Location: Omaha

NetID: NOT SET

Balance: $0.00

Credits: 0

GPA: 0.00

UNO STUDENT 2 TEST OK

Name: Cain Marko

NUID: 68345919

Campus Location: Lincoln

MyUNLID: cain.marko

Balance: $6459.21

Credits: 21

GPA: 2.00

UNL STUDENT 1 TEST OK

Name: Carl Lykos

NUID: NOT SET

Campus Location: Lincoln

MyUNLID: NOT SET

Balance: $0.00

Credits: 0

GPA: 0.00

UNL STUDENT 2 TEST OK

Name: Scott Summers

NUID: 10000000

Campus Location: Omaha

NetID: ssummers

Position: Receptionist

Wage: $13.50

UNO STAFF 1 TEST OK

Name: Henry McCoy

NUID: NOT SET

Campus Location: Omaha

NetID: NOT SET

Position: NOT SET

Wage: NOT SET

UNO STAFF 2 TEST OK

Name: Victor Creed

NUID: 99999999

Campus Location: Lincoln

MyUNLID: victor.creed

Position: Yard Services

Wage: $9.00

UNL STAFF 1 TEST OK

Name: Mortimer Toynbee

NUID: NOT SET

Campus Location: Lincoln

MyUNLID: NOT SET

Position: NOT SET

Wage: NOT SET

UNL STAFF 2 TEST OK

Name: Charles Xavier

NUID: 58355934

Campus Location: Omaha

NetID: cxavier

Department: Computer Science

Title: Professor

Salary: $700000.00

UNO FACULTY 1 TEST OK

Name: Bobby Drake

NUID: NOT SET

Campus Location: Omaha

NetID: NOT SET

Department: NOT SET

Title: NOT SET

Salary: NOT SET

UNO FACULTY 2 TEST OK

Name: Erik Lehnsherr

NUID: 98658472

Campus Location: Lincoln

MyUNLID: erik.lehnsherr

Department: History

Title: Assistant Professor

Salary: $23660.00

UNL FACULTY 1 TEST OK

Name: Arkady Rossovich

NUID: NOT SET

Campus Location: Lincoln

MyUNLID: NOT SET

Department: NOT SET

Title: NOT SET

Salary: NOT SET

UNL FACULTY 2 TEST OK

**Assignment Questions**

Provide the answers to these questions in your submission directory within a file called Assignment1Questions.

1. In creating these classes what advantage to we gain by utilizing the Object Oriented Programming concept of Inheritance?

2. Come up with your own hierarchy design that would benefit from Inheritance. Your hierarchy must have at least 3 levels and be made up of at least 7 classes. You do not have to write any code. Describe the classes and how they would relate to each other as well as some of the data members and methods the classes would have. Explain why your hierarchy makes sense. Your design could be something that relates to your particular field of study or hobbies. Do not use examples used in class or discussed in the text book.