Note:

1. Projects can be completed in groups (max 4 per group) but submission onto WebCourses is individual. Once again, don’t copy anybody’s work nor give your code to anybody unless you **seriously** worked together on the project.

**You will not receive any credit if you don’t submit your .java file by the deadline!**

1. Please Submit one .java file. It should look like this:

**public** **class** Project2 {

**public** **static** **void** main(String[] args){

//Test code goes here

}

}

//---------------------------

**class** Abc{

}

//---------------------------

**class** Xyz*{*

}

1. If you work with a classmate(s), your .java file must contain the following comment:

/\*

* Project 3
* Names (first and last names) of all students who worked together on the project
* (optional) Add anything that you would like the TA to be aware of

\*/

Example:

/\*

* Project 3
* Hatim Boustique, Ericka Edwards and Jamal Dubois

\*/

**public** **class** Project3 {

**public** **static** **void** main(String[] args){

//Test code goes here

}

}

//---------------------------

**inteface** Int1 {

}

//---------------------------

**class** Abc{

}

//---------------------------

**class** Xyz*{*

}

**Project 3 statement**

Please read this entire statement carefully before you start doing anything…

This project is similar to the previous one. It involves implementing a simple university personnel management program. The program contains two different kinds of objects: students and faculty. For each object, the program stores relevant information such as university ID, name, etc. Different information is stored depending on the type of the object. For example, a student has a GPA, while a faculty has a title and department (professor, mathematics).  
  
For each of any class data member, your program must include the **getters** and the **setters**, and each class must include at least **two constructors**. The goal of this Project is to demonstrate the use of inheritance, abstract classes or interfaces, abstract methods and method overriding.

Student class has:

* Full name
* ID
* Gpa
* Number of credit hours currently taken

Faculty class has:

* Full name
* ID
* Department (mathematics, engineering, arts and science)
* Rank (professor, adjunct)

Students in this college pay $236.45 per credit hour in addition to a $52 administrative fee. Your code should generate a tuition invoice **( a method within the class Student**). Note that students get a 25% off total payment if their gpa is greater or equal to 3.85.

**Both classes Faculty and Student inherit from the abstract class Person. The abstract class Person has what’s common to both Faculty and Student, and it is left to you (the programmer) to come out with an abstract method that you implement in your classes Faculty and Student, say void printInfo();**

**You may change the class Person to an interface to be implemented by Faculty and Student. Either way is fine!**

**Your code should handle any Input Mismatch Exceptions. In addition, you code must have a user-defined exception called IdException that handles the faculty and student ids entered by the user. Note that any id has to be of the form LetterLetterDigitDigitDigitDigit like el7894. See sample run below.**

**Test your code with one array of size 100 (or ArrayList) of type Person. The sample run below should give you a clear idea about how your code should run. The user’s entry is marked in bold so you can tell what your code should display to the screen and what the user enters.**

Please note well that:

1. Your code should run exactly as shown on the sample run below (However, the TA will not deduct points because you skipped two lines instead of three or your tuition invoice has 56 hyphens instead of 63! ).
2. When asked to enter the enter the faulty department, **matheMatics** and **MathematiCs** are considered to be the same, and your program should display ***mathematics*** if faculty information is to be displayed to the screen. However, if the user enters ***Mathematics department***, then this is an invalid entry.
3. The university ID has no required form so you may choose to enter anything to be the ID.

Welcome to my Personal Management Program

Choose one of the options:

1. Add a new Faculty member
2. Add a new Student
3. Print tuition invoice for a student
4. Print information of a faculty
5. Exit Program

Enter your selection: **2**

Enter the student’s info:

Name of Student: **Julia Alvarez**

ID: **ju1254**

Gpa: **3.26**

Credit hours: **7**

Thanks!

Choose one of the options:

1. Add a new Faculty member
2. Add a new Student
3. Print tuition invoice for a student
4. Print information of a faculty
5. Exit Program

Enter your selection: **2**

Enter the student’s info:

Name of Student: **Matt Jones**

ID: **ma0258**

Gpa: **2.78**

Credit hours: **0**

Thanks!

1. Add a new Faculty member
2. Add a new Student
3. Print tuition invoice for a student
4. Print information of a faculty
5. Exit Program

Enter your selection: **a**

Invalid entry- please try again

1. Add a new Faculty member
2. Add a new Student
3. Print tuition invoice for a student
4. Print information of a faculty
5. Exit Program

Enter your selection: **1**

Enter the faculty’s info:

Name of the faculty: **John Miller**

ID: **jo7894**

Rank: **Instructor**

Sorry entered rank (Instructor) is invalid

Rank: **Professor**

Department: **Engineering**

Thanks!

1. Add a new Faculty member
2. Add a new Student
3. Print tuition invoice for a student
4. Print information of a faculty
5. Exit Program

Enter your selection: **3**

Enter the student’s id: **ju1254**

Here is the tuition invoice for Julia Alvarez :

---------------------------------------------------------------------------

Julia Alvarez ju1254

Credit Hours:7 ($236.45/credit hour)

Fees: $52

Total payment: $1,707.15 ($0 discount applied)

1. Add a new Faculty member
2. Add a new Student
3. Print tuition invoice for a student
4. Print information of a faculty
5. Exit Program

Enter your selection: **3**

Enter the student’s id: **eri856**

Sorry-student not found!

1. Add a new Faculty member
2. Add a new Student
3. Print tuition invoice for a student
4. Print information of a faculty
5. Exit Program

Enter your selection: **4**

Enter the faculty’s id: **jo8578**

Sorry jo8578 doesn’t exist

1. Add a new Faculty member
2. Add a new Student
3. Print tuition invoice for a student
4. Print information of a faculty
5. Exit Program

Enter your selection: **4**

Enter the faculty’s id: **JO7894**

Faculty found:

---------------------------------------------------------------------------

John Miller

Engineering Department, Professor

---------------------------------------------------------------------------

1. Add a new Faculty member
2. Add a new Student
3. Print tuition invoice for a student
4. Print information of a faculty
5. Exit Program

Enter your selection: **1**

Enter the faculty’s info:

Name of the faculty: **Erika J. Jones**

ID: **e7894**

Sorry Invalid id format-It has to be LetterLetterDigitDigitDigitDigit

1. Add a new Faculty member
2. Add a new Student
3. Print tuition invoice for a student
4. Print information of a faculty
5. Exit Program

Enter your selection: **5**

Goodbye!

OPTIONAL: You may try having something like:

**class** Faculty **extends** Person **implements** SomeInterface

**class** Student **extends** Person **implements** SomeInterface

But the issue is how can you have one array to manage both the Faculty and Student objects? If you are interested, please give me a call (321) 900-1696 and we’ll chat about a possible solution.