Name: _____

13c – XML lab

Overview:

The scope of what XML covers is great. This lab gives you practice understanding the XML structure and writing code to process the XML and treat them as objects.

Setup:

Starter code that parses an XML file StudentParser.java, and the XML file, students.html, are supplied in MyCourses. Run that program to see how they work together.

Requirements for this lab:

A second XML file, studentCourses.xml, is supplied. The original code described in setup above can be modified to handle the new XML file. The student XML file is not going to be about grades and GPA, but for contact information. Here are the requirements to read the new XML file. Look at the XML file while you read these requirements:

1. Change the input XML filename from students.xml to studentCourses.xml.
2. The GPA is to be removed from the code.
3. The student's first and last names are to be placed into a class you write named Person. The Person class has attributes for First and Last names. Create a constructor and accessors for first and last names.
4. The new XML file now contains Courses for the student. Create a Course class that has the attributes: name and course number, for one course. Create a toString() method to return the course number and course name in a string, suitable for printing as the example output shows.
5. Write a Student class with the following attributes:
    a. Student id
    b. Person object
    c. ArrayList of Courses

    The Student class has a constructor that takes an id, Person, and ArrayList of Courses as parameters.
    This class also has a toString() that returns the Id, person's name, and a list of courses all formatted into one string for this one student ready for printing as shown in the example output.
6. The main class, StudentParser, needs to have this design and new attributes:
    a. An ArrayList to hold Student objects.
    b. Constructor does not change from supplied code.
    c. A parse method that works similar to the sample.
        For each <student> entry:
        i. Get the id
        ii. Get the last name
        iii. Get the first name

iv. Create a person object with the first and last names
v. Create an ArrayList of Course objects
vi. Parse through the courses for each student, getting the course name and number. Create a Course object from these two pieces of information (Note: use /@(attributeName) to get attributes)
Example: <test>
<intest name="nameHere" />
</test>
path to use: "/test/intest/@name"->this returns "nameHere"
vii. Add the Course object to the ArrayList of course objects
viii. After all the courses have been added to the ArrayList, create a Student object that contains the Person object, course ArrayList, and the student id.
ix. Add this student object to an array list of students in the StudentParser class.

d. A printReport() method is needed that loops through the array list of students, and prints the student object. The toString() of the student class is used to print the preformatted output.
e. The main() method creates an object of this StudentParser class.
It calls the parse method, passing in the XML filename.
It calls the printReport() method.

Sample Output:

$ **java StudentParser**
*** Students Listing ***
ID: 12345 Matthew Bettinger
UNDW-333: Underwater Basket Weaving
ISTE-200: Java for Programmer
ID: 12355 Andrew Collins
UNDW-333: Underwater Basket Weaving
ISTE-200: Java for Programmer
ISTE-140: Web 1
ID: 12365 Linda Robinson
ISTE-121: Programming for the Info. Tech. Domain II

Sample of XML file: (this is not the entire XML file studentCourses.xml)

```
<student>
    <id>12345</id>
    <person>
      <lastperson>Bettinger</lastperson>
      <firstperson>Matthew</firstperson>
    </person>
    <courses>
      <course name="Underwater Basket Weaving" coursenumber="UNDW-333" />
      <course name="Java for Programmer" coursenumber="ISTE-200" />
    </courses>
  </student>
```

Instructor / TA: _____