1 Submission Instructions

Create a document using your favorite word processor and type your exercise solutions. At the top of the document be sure to include your name and the homework assignment number, e.g. HW1. Convert this document into **Adobe PDF format** and name the PDF file *<asuriteid>.pdf* where *<asuriteid>* is your ASURITE user id (for example, my ASURITE user id is *vnaray15* so my file would be named *vnaray15.pdf*). To convert your document into PDF format, Microsoft Office versions 2008 and newer will export the document into PDF format by selecting the proper menu item from the File menu. The same is true of Open Office and Libre Office. Otherwise, you may use a freeware PDF converter program, e.g., *CutePDF* is one such program.

Next, create a folder named *<asuriteid>* and copy *<asuriteid>.pdf* to that folder. Copy any Java source code files to this folder (note: Java source code files are the files with a *.java* file name extension; **do not** copy the *.class* files as we do not need those).

Next, compress the *<asuriteid>* folder creating a **zip archive** file named *<asuriteid>.zip*. Upload *<asuriteid>.zip* to the Homework Assignment 1 **link** by the assignment deadline. **The deadline is** in course schedule **.** Consult the online syllabus for the late and academic integrity policies.

Note: not all of these exercises will be graded, i.e., random ones will be selected for grading.

2 Learning Objectives

1. Use the Integer and Double wrapper classes.
2. Declare and use ArrayList class objects.
3. Write code to read from, and write to, text files.
4. Write an exception handler for an I/O exception.
5. Write Java classes and instantiate objects of those classes.
6. Read UML class diagrams and convert the diagram into Java classes.
7. Identify and implement dependency, aggregation, inheritance, and composition relationships.

3 ArrayLists

3.1 Write code that creates an ArrayList<Integer> object named *list* and fills *list* with these numbers (using one or a pair of **for** or **while** loops):

```
0  1  2  3  4  0  1  2  3  4
```

3.2 Consider the ArrayList<Integer> object named *list* containing these Integers:

```
list = { 1, 2, 3, 4, 5, 4, 3, 2, 1, 0 }
```

What are the contents of *list* after this loop completes?

```
for (int i = 1; i < 10; ++i) {
    list.set(i, list.get(i) + list.get(i-1));
}
```

3.3 Write an **enhanced for loop** that counts how many numbers in an ArrayList<Integer> object named *list* are negative. Print the count after the loop terminates.

3.4 Write a method named *arrayListSum*() that has one parameter *pList* which is an object of the class ArrayList <Integer>. The method shall return the sum of the elements of *pList*.

3.5 Write a method named named *arrayListCreate*() that has two int parameters *pLen* and *pInitValue*. The method shall create and return a new ArrayList<Integer> object which has exactly *pLen* elements. Each element shall be initialized to *pInitValue*.

3.6 Write a void method named *insertName*() that has two input parameters: (1) *pList* which is an object of ArrayList

<String> where each element of *pList* is a person's name; and (2) a String object named *pName*. Assume the names in *pList* are sorted into ascending order. The method shall insert *pName* into *pList* such that the sort order is maintained.

3.7 Write a void method named *arrayListRemove*() which has two parameters: *pList* is an object of the ArrayList <Integer> class and *pValue* is an int. The method shall remove all elements from *pList* that are equal to *pValue*.

4 Text File Input/Output

4.1 Explain what happens if you try to open a file for reading that does not exist.

4.2 Explain what happens if you try to open a file for writing that does not exist.

4.3 **Name your source code file *Hw1_1.java*.** Write a program that prompts the user for the name of a Java source code file. The program shall read the source code file and output the contents to a new file named the same as the input file, but with a *.txt* file name extension (e.g., if the input file is *foo.java* then the output file shall be named *foo.java.txt*). Each line of the input file shall be numbered with a line number (formatted as shown below) in the output file. For example, if the user enters *Hw1_1.java* as the name of the input file and *Hw1_1.java* contains:

```
//*************************************************************
// CLASS: Hw1_1 (Hw1_1.java)
//*************************************************************
public class Hw1_1 {
    public static void main(String[] pArgs) {
    }
    public Hw1_1() {
    }
}
```

then the contents of the output file *Hw1_1.java.txt* would be:

```
[001] //*************************************************************
[002] // CLASS: Hw1_1 (Hw1_1.java)
[003] //*************************************************************
[004] public class Hw1_1 {
[005]     public static void main(String[] pArgs) {
[006]     }
[007]     public Hw1_1() {
[008]     }
[009] }
```

Hint: to print an integer as shown above in a field of width 3 with leading 0's use the printf() method with a format specifier of %03d.

5 Exceptions and Exception Handling

5.1 Explain the difference between throwing an exception and catching an exception, i.e., explain what happens when an exception is thrown and when one is caught?

5.2 Explain what a checked exception is. Give one example.

5.3 Explain what an unchecked exception is. Give one example.

5.4 Which type of uncaught exceptions must be declared with the throws reserved word in a method header?

5.5 Why don't you need to declare that your method might throw an IndexOutOfBoundsException?

5.6 Is the type of the exception object always the same as the type declared in the catch clause that catches it? If not, why not?

5.7 What is the purpose of the finally clause? Give an example of how it can be used.

5.8 Which exceptions can the *next*() and *nextInt*() methods of the Scanner class throw? Are they checked exceptions or unchecked exceptions?

6 Objects and Classes

6.1 Explain how an instance method differs from a class method (static method).

6.2 Explain what happens if we write a class named *C* but do not implement any constructors in *C*.

6.3 (a) In a static method, it is easy to differentiate between calls to instance methods and calls to static methods. How do you tell them apart? (b) Why is it not as easy for methods that are called from an instance method?

6.4 Explain what happens when this application runs and why.

```
public class C {
    private int x;
    private String s;
    public static void main(String[] pArgs) {
        new C();
    }
    public C() {
        x = s.length();
        System.out.println("x = " + x);
    }
}
```

6.5 Write the declaration for a class named *C* that declares: (1) a private int instance variable named *mX*; (2) a private int class variable named *mY* initialized to 0; (3) a private int class constant named *A* which is equivalent to 100; (4) a public int class constant named *B* which is equivalent to 200; (5) public accessor and mutator methods for *mX* named *getX*() and *setX*(); (6) public accessor and mutator methods for *mY* named *getY*() and *setY*(); (7) a constructor that has one int input parameter named *pX* which calls *setX*() to initialize *mX* to *pX*; (8) a default constructor that calls *C*(int) to initialize *mX* to -1.

6.6 Continuing with the previous exercise, write the declaration for a class named *Main* that implements the *main*() method. Within *main*() suppose we wish to instantiate a *C* object named *cObj1* calling the default constructor. Write the code to do this.

6.7 Continuing, write the code to instantiate another *C* object named *cObj2* calling the second constructor to initialize the *mX* instance variable to 10.

6.8 Continuing, within *main*(), are the following statements legal, i.e., do they compile? If so, explain what happens when the statement is executed. If not, explain why the statement is illegal.

(a) `int a1 = C.mX;`
(b) `int a2 = C.mY;`
(c) `int a3 = C.A;`
(d) `int a4 = C.B;`
(e) `cObj1.C(20);`
(f) `int a5 = cObj1.getX();`
(g) `cObj1.setX(20);`
(h) `cObj2.setX(cObj1.getX());`
(i) `int a6 = C.getX();`
(j) `C.setX(20);`
(k) `int a7 = cObj1.getY();`
(l) `cObj1.setY(20);`

(m) `int a8 = C.getY();`

(n) `C.setY(20);`

6.9 Continuing, suppose we add these two methods to the declaration of class *C*. For each assignment statement, if it is legal (compiles) explain what happens when the statement is executed. For each assignment statement that is illegal (syntax error) explain why the code is illegal.

```
public void f() {
    mX = 0;
    mY = 0;
}
public static void g() {
    mX = 0;
    mY = 0;
}
```

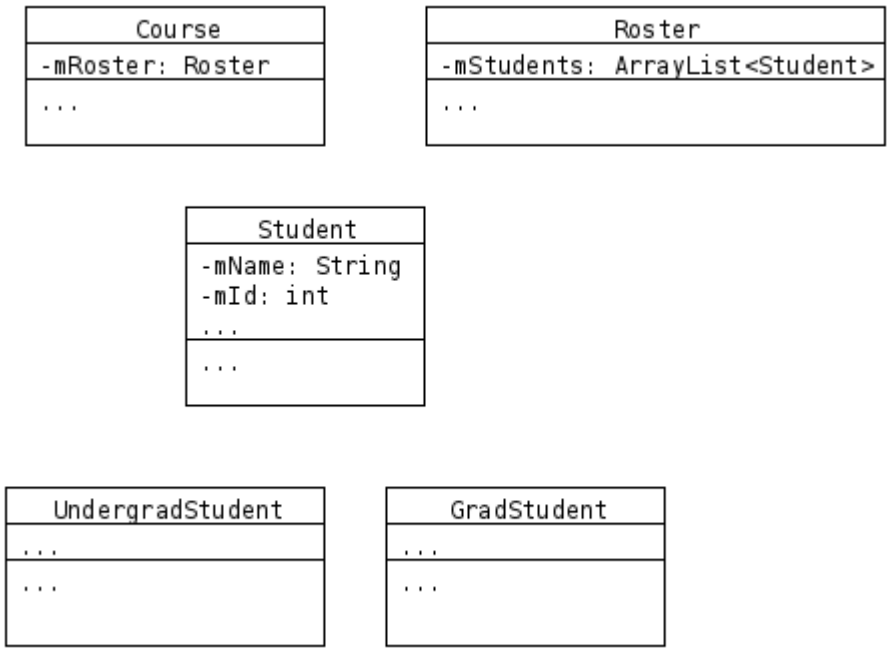
7 Object Oriented Design and UML Class Diagrams

7.1 Below is the code for a Java class named *C*. Draw the UML class diagram that corresponds to this code. There are several free and open-source tools for drawing UML diagrams. Two reasonably simple and cross-platform (Windows, Mac, Linux) tools are Umlet (http://www.umlet.com/) and Violet (http://alexdp.free.fr/violetumleditor). I have been using Umlet for the diagrams I am drawing and I highly recommend it as it is very easy to quickly draw a nice-looking class diagram. Using Umlet, you can export your diagram as an image file by selecting File | Export As... on the main menu. I recommend you export the diagram as a .png image and then copy-and paste the .png image into your word processing document.

```
public class C {
    public static final int CONST1 = 100;
    private int mX;
    private double mY;
    private String mZ;
    public C() { this(0, 0, ""); }
    public C(int pX, double pY, String pZ) { setX(pX); setY(pY); setZ(pZ); }
    public int getX() { return mX; }
    public int getY() { return mY; }
    private String getZ() { return mZ; }
    public void setX(int pX) { mX = pX; }
    public void setY(int pY) { mY = pY; }
    private void setZ(int pZ) { mZ = pZ; }
}
```

7.2 On the next page is a UML class diagram that shows several classes that are related in various ways. **(a)** What type of class relationship, if any, exists between *Course* and *Roster*? If there is a relationship, modify the diagram to indicate the relationship and label each end of the relationship with a multiplicity. **(b)** What type of class relationship, if any, exists between *Roster* and *Student*. If there is a relationship, modify the diagram to indicate the relationship and label each end of the relationship with a multiplicity. **(c)** What type of class relationship, if any, exists between Student and UndergradStudent? If there is a relationship, modify the diagram to indicate the relationship and label each end of the relationship with a multiplicity. **(d)** What type of class relationship, if any, exists between Student and GradStudent? If there is a relationship, modify the diagram to indicate the relationship and label each end of the relationship with a multiplicity. **(e)** What type of class relationship, if any, exists between UndergradStudent and GradStudent? If there is a relationship, modify the diagram to indicate the relationship and label each end of the relationship with a multiplicity.

Note: Draw this diagram in a UML diagramming tool such as Umlet or Violet and copy-and-paste your completed diagram into your word processing document.


Course
-mRoster: Roster
...
Roster
-mStudents: ArrayList<Student>
...
Student
-mName: String
-mId: int
...
...
UndergradStudent
...
...
GradStudent
...
...