James J. Kim
Homework 1

3.1 Write code that creates an ArrayList 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

```
import java.util.ArrayList;

public class Homework1 {

    public static void main(String[] args) {

        // Instantiate a new ArrayList object
        ArrayList<Integer> list = new ArrayList<>();

        // Iterate and fill the ArrayList
        for (int i = 0; i < 5; i++) {
            list.add(i);
        }

        // This is ugly to write this again... but quick easy way
        for (int i = 0; i < 5; i++) {
            list.add(i);
        }

        // DBEUG: Print out the ArrayList for debugging
        for (Integer intVar : list) {
            System.out.print(intVar);
        }
    }
}
```

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

```
public int arrayListSum(ArrayList pList) {

    // hold the total in the pList object
    int sum = 0;

    for (int i = 0; i < pList.size(); i++) {

       // Cast the element in the ArrayList object to integer
       sum += (Integer)pList.get(i);
    }

    // Return the sum
    return sum;
 }
```

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 object which has exactly pLen elements. Each element shall be initialized to pInitValue

```
public ArrayList arrayListCreate(int pLen, int pInitValue) {

    // new ArrayList of int
    ArrayList<Integer> newArrayList = new ArrayList<>();

    // fill the new ArrayList with the init value
    for (int i = 0; i < pLen; i++) {
       newArrayList.add(pInitValue);
    }

    return newArrayList;
 }
```

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:

File Hw1_1.java attached.

5.3 Explain what an unchecked exception is. Give one example

Unchecked exception can occur due to the bug in the code or other abnormal condition. One example of unchecked exception is IndexOutOfBounds exception. This can occur when attempting to access an element in an array that is outside the bounds of last element in the array.

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

If your program crashes due to IndexOutOfBound, you want to be able to read through the stack trace and determine the root cause.

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?

This is not necessarily true. The exception object can be a superclass of the class that's caught in the catch clause. For example, in the method declaration, you can set it to throw IOException and catch FileNotFoundException in the catch clause.

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?

When calling the static method, you'll need to use the ClassName.staticName() and not the instanceName.instanceMethod(). The method or property belongs to the class and not the instance variable hence you will use the class name.

(b) Why is it not as easy for methods that are called from an instance method?

Static methods and data members do not belong to the instance of the class but the class itself. Therefore, you can't call a static method from an instance class.

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.

C cObj2 = new C(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; *Not compilable because mX is not accessible as its a private data member and its also an instance variable.*

(b) int a2 = C.mY; *Not compilable because mY is not accessible as its a private data member and its also an instance variable.*

(c) int a3 = C.A; *Not compilable because 'A' data member is not accessible outside of the class its declared as private.*

(d) int a4 = C.B; *Compilable and this will work.*

(e) cObj1.C(20); *Not compilable - there is no method name C for class C*

(f) int a5 = cObj1.getX(); *Compilable and this will work. Using the accessor to retrieve a private data member.*

(g) cObj1.setX(20); *Compilable and this will work. Using the accessor to set a private data member.*

(h) cObj2.setX(cObj1.getX()); *Compilable and this will work. Using the accessor to retrieve a private data member and to use that return value to set another.*

(i) int a6 = C.getX(); *Not compilable; getX() is an instance method and not a class method.*

(j) C.setX(20); *Not compilable; setX() is an instance method and not a class method.*

(k) int a7 = cObj1.getY(); *Compilable and this will work. Using the accessor to retrieve a private data member.*

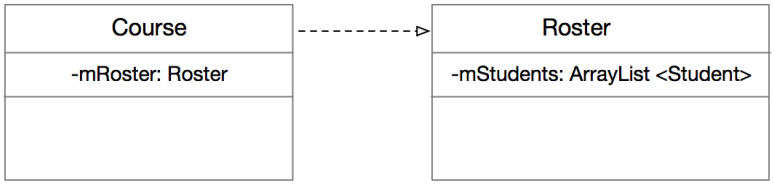
(l) cObj1.setY(20); *Compilable and this will work. Using the accessor to set a private data member.*

(m) int a8 = C.getY(); *Not compilable; getY() is an instance method and not a class method.*

(n) C.setY(20); *Not compilable; setY() is an instance method and not a class method.*

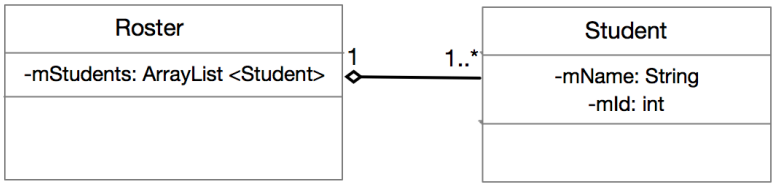
7.2 On the next page is a UML class diagram that shows several classes that are related in various ways.
(a) What type of class relationship, if any, exists between Course and Roster? If there is a relationship, modify the diagram to indicate the relationship and label each end of the relationship with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

The Course class is considered to be dependent on the Roster class.



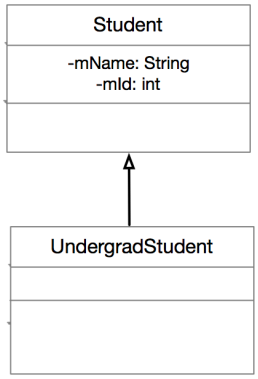
(b) What type of class relationship, if any, exists between Roster and Student. If there is a relationship, modify the diagram to indicate the relationship and label each end of the relationship with a multiplicity.

This is considered to be an aggregation relationship. Roster class can have multiple instances of Student object inside the ArrayList



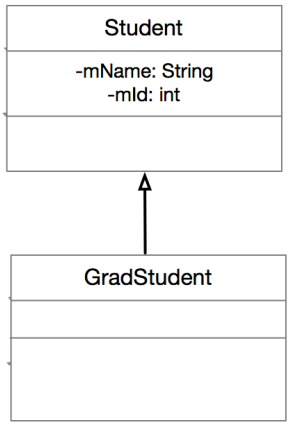
(c) What type of class relationship, if any, exists between Student and UndergradStudent? If there is a relationship, modify the diagram to indicate the relationship and label each end of the relationship with a multiplicity.

UndergradStudent class is considered to be a subclass of Student class. UndergradStudent inherits all the attributes of a Student class because UndergradStudent IS-A Student.



(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.

GradStudent is considered to be a type of Student. Student is considered to be a superclass or parent class of GradStudent and GradStudent is considered to be a subclass of a Student class.



(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.

The only relationship between UndergradStudent and the GradStudent class is the superclass they inherit from. They are both considered to be a STUDENT but GradStudent is NOT a UndergradStudent.