Objectives:

As a programmer, you will be expected to understand good coding practice and logical structures. For this project ***you must show mastery of***:

- Proper code layout
- Variable declaration and initialization
- Proper library importing
- Constant declaration and initialization
- Collections (arrays)
- Method overloading
- toString
- Modularization
- Enums
- instanceof
- Main method
- IntelliJ IDEA usage
- Random
- Program sequence, selection, & looping
- Cohesion
- UML
- JavaDoc
- Interfaces
- Abstract class
- Input validation/sanitization
- Good programming practices

Lab 10

All of the chapter exercises plus:

(CarbonFootprint Interface: Polymorphism) Using interfaces, as you learned in this chapter, you can specify similar behaviors for possibly disparate classes. Governments and companies worldwide are becoming increasingly concerned with carbon footprints (annual releases of carbon dioxide into the atmosphere) from buildings burning various types of fuels for heat, vehicles burning fuels for power, and the like. Many scientists blame these greenhouse gases for the phenomenon called global warming. Create three small classes unrelated by inheritance—classes Building, Car and Bicycle. Give each class some unique appropriate attributes and behaviors that it does not have in common with other classes. Write an interface CarbonFootprint with a getCarbonFootprint method. Have each of your classes implement that interface, so that its getCarbonFootprint method calculates an appropriate carbon footprint for that class (check out a few websites that explain how to calculate carbon footprints). Write an application that creates objects of each of the three classes, places references to those objects in ArrayList< CarbonFootprint>, then iterates through the ArrayList, polymorphically invoking each object's getCarbonFootprint method. For each object, print some identifying information and the object's carbon footprint using toString. Create an abstract class that one of your classes can extend in addition to the implemented interface.