Tutorial 1: Type hierarchy (1)

Exercise 1. Define and implement a subtype of `java.util.ArrayList` called `MaxMinIntList` that provides methods to return the smallest (`min()`) and largest (`max()`) elements of the list. Be sure to define the rep invariant and abstraction function, and to implement `repOk()`. You need to code `main` function to test.

Exercise 2: Consider a type `Counter` with the following operations:

```
/**
 * @effects Makes this contain 0
 */
public Counter()

/**
*
* @effects Returns the value of this
*/
public int get()

/**
* @modifies this
* @effects Increments the value of this
*/
public void incr()
```

Complete the specification of `Counter` by providing the overview section. Be sure to identify all properties of `Counter` objects. You need to code `main` function to test.

Exercise 3. Inheritance

This exercise uses the `Vehicle` type hierarchy example that was used in the lecture.

1. Update the two classes `Bus` and `Car` so that their weight constraints are as follow:

   1.1. `Bus.weight` is in the range [5000.0, 20000.0] (kgs)

   1.2. `Car.weight` is in the range [1000.0, 2000.0] (kgs)

2. Update the two classes `Bus` and `Car` so that they now have the following constraints on the length dimension:

   2.1. `Bus.length` is in the range [4.0, 10.0] (meters)

   2.2. `Car.length` is in the range [1.5, 3.5] (meters)

3. Update class `Vehicle` to have a new attribute called `registrationNumber`. Based on your practical understanding of this attribute, decide a suitable data type and restrictions for it. *Note:* you must update and/or define the operations that are relevant to the new attribute.

4. Update the two classes `Bus` and `Car` so that they each have different restrictions for the attribute `registrationNumber` from the restrictions defined in the class `Vehicle`. For example, if `Vehicle.registrationNumber` can contain up to 12 alpha-numerical characters then `Bus.registrationNumber` and `Car.registrationNumber` could only contains up to 8 and (respectively) 6 such characters.
5. Update the three classes `Vehicle`, `Bus` and `Car` so that the `toString()` method can be removed from `Bus` and `Car`, and that the inherited `toString()` method from the class `Vehicle` now provides the accurate class label for not only `Vehicle` but also for `Bus` and `Car`. *Hint:* In Java, you can use the following statement in a method to get the actual (run-time) type of the object that carries that method: `this.getClass().getSimpleName()`.

You need to code `main` function to test.