CS 251
Lab Exercise 08

Main topics: Interfaces
Abstract methods
Polymorphism

Exercise

This week we will be practicing writing a class which implements an interface.

Getting Started To start this exercise, you should:

1. Open eclipse and start a new Java project named `Lab08`
2. Add a Class (named `Nat`) to this project, and copy the contents of the `Nat.java` file provided into it.
3. Add a Class (named `Mod8`) to this project, and copy the contents of the `Mod8.java` file provided into it.
4. Add a Class (named `Oct`) to this project, the contents of which you will be writting from scratch.
5. Add a Class (named `DriverL8`) to this project, and copy the contents of the `DriverL8.java` file provided into it.

Requirements

Nat.java A very simple class which models some of the functionality of a Natural number.
The Natrual numbers are the non-negative Integers: 0, ...
This class is complete and must not be modified.

1. Why is `setN()` protected?

Mod8.java A very simple interface which implies a the cyclic set of Natural numbers [0, 7].
This class is complete and must not be modified.

1. Why does the notion of an additive inverse not make sence for Natural numbers, but does make sence for the cyclic set [0, 7]?

Oct.java A very simple class which extends the Nat class and implements the Mod8 interface.
This class you must write, such that:

1. You include default, specifying and copy constructors
2. You override Nat's mutator.
3. You override Nat's clone method
4. You override Nat's equals method
5. You implement Mod8's inverse method
   For any `x` in [0, 7], its unique inverse `i` has the property that `(x + i) % 8 == 0`
6. You override Nat's increment, decrement and addition methods so that the underlying int value is always in the cyclic range [0, 7].

DriverL8.java A simple driver class to test all of the above classes / interface.
This class is complete and must not be modified.

1. Identify how / where Polymorphism is being utilized.

If you wrote your Oct class correctly then your output will look like:

```
n1 = 0
n2 = 7
n3 = 7
n4 = 0
n5 = 0

no1 = 0
no2 = 7
no3 = 7
no4 = 0
no5 = 2

Good: Oct can not equal Nat
Good: Oct.clone() works
Good: Oct(Oct o) works

n2 + n3 = 14

n1 = 0
n2 = 8
n3 = 0

no5 inverse = 6
no2 + no5 = 1
no2 - no5 = 5

no1 = 7
no2 = 0
no3 = 0
```

Once you have completed the requirements:

1. Make sure that your program runs without errors or warnings.
2. Run your program enough times to verify its correctness.
3. If it runs correctly, then see your TA for a check-off.