Observer Pattern Exercise

Download and study the Alarm Clock application. The responsibilities of the `AlarmClock` and `Person` classes should be clear. Play with this code and make sure you understand it.

Now look at the `AlarmApplication` class. There's a lot of control code in this class (it has God Class tendencies), which is never a good thing. It instantiates `Person` and `AlarmClock`, sets the alarm, sends the person to bed, runs the alarm clock for a day, checking after each tick if the alarm has been reached. If so, it wakes the person up.

In performing the following exercises, think about the pattern and what you are doing.

1. Make sure you understand the Observer design pattern to start with. Now apply Observer to handle the interaction between `Person` and `AlarmClock`. Some things to note on this:

   - `AlarmClock` will have a list of observers and inform them whenever its alarm goes off.
   - `Person` will be given a reference to an `AlarmClock` object, and will have an update() method that does the right thing whenever the alarm goes off.
   - The `AlarmApplication` class will only have to instantiate the other classes, set the alarm and invoke `tick` in a loop.
   - Implement your own `Observer` interface and `Observable` abstract class — don't use the ones provided in the Java libraries.

2. Extend `AlarmApplication` so that it creates three `Person` objects, all to be woken at the same time. This should be easy to do if you have done Part 1 correctly.

3. Extend the `Person` class so that it stores the wake-up time as well. Whenever the alarm time is set in the `AlarmClock`, it should be updated in all the observing `Person` objects as well. Update `AlarmApplication` to verify this functionality.

4. Whichever model you used in Part 3, push or pull, refactor your code to use the other model.

5. Create a new class called `Cron` (unrelated to `Person`) that performs some task when a certain time is reached. If you've applied Observer correctly, it should be trivial to add this new type of observer.

6. If you have time, play around with some of the other issues we looked at around the Observer pattern.