1. **Choosing fields** For each of the following kinds of values, choose appropriate fields to represent the value (imagine you are trying to implement a class that represents the value). Try to come up with two alternate sets of fields that could represent each kind of value.

   (a) weight

   (b) temperature

   (c) time of the day

   (d) day of the year

2. **Default constructor**

   (a) Suppose that a `Temperature` is represented as a floating point value in degrees Celcius. Implement a default (no argument) constructor.

   (b) Suppose that a `TimeOfDay` is represented as an integer hour and an integer minute. Implement a default (no argument) constructor.

3. **Custom constructor**

   (a) Suppose that a `Temperature` is represented as a floating point value in degrees Celcius. Implement a custom constructor that initializes the temperature given a value in degrees Celcius.

   (b) Suppose that a `TimeOfDay` is represented as an integer hour and an integer minute. Implement a custom constructor that initializes a time given an hour and a minute.

4. **Copy constructor**

   (a) Suppose that a `Temperature` is represented as a floating point value in degrees Celcius. Implement a copy constructor that initializes the temperature given another `Temperature` reference.

   (b) Suppose that a `TimeOfDay` is represented as an integer hour and an integer minute. Implement a copy constructor that initializes a time given another `TimeOfDay` reference.

5. **Implement a set method**

   (a) Suppose that a `Temperature` is represented as a floating point value in degrees Celcius. Implement a `set` method that sets the value of a temperature given a value in degrees Celcius.

   (b) Suppose that a `TimeOfDay` is represented as an integer hour and an integer minute. Implement a `set` method that sets the value of a time given an hour and a minute.

6. **toString**

   Implement a `toString` method for `Temperature` and `TimeOfDay`.

7. **equals**

Consider a `Card` class that represents a standard playing card. Every `Card` object has a `Rank` and a `Suit`. Complete the `equals` method for `Card`. You may assume that the `Rank` and `Suit` classes both have an `equals` method.

```
public class Card implements Comparable<Card> {

   private Rank rank;
   private Suit suit;

   /**
    * Compares this playing card to the specified object. The result
    * is <code>true</code> if and only if the argument is a
    * <code>Card</code> with the same rank and suit as this card.
    *
    * @param obj
    *            The object to compare this Card against.
    * @return true if the given object is a
    *         Card equal to this playing card,
    *         false otherwise.
    */
        @Override
        public boolean equals(Object obj) {
        if (                                        ) {



        }
        if (                                        ) {



        }
        if (                                        ) {



        }
}
```