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**

   Implement a default constructor for two of the kinds of values from Question 1.

3. **Custom constructor**

   Implement a custom constructor for two of the kinds of values from Question 1.

4. **Copy constructor**

   Implement a copy constructor for two of the kinds of values from Question 1.

5. **Implement a set method**

   Implement a `set` method for two of the kinds of values from Question 1.

6. **toString**

   Implement a `toString` method for two of the kinds of values from Question 1.

7. **equals**

Consider the `Card` class from Lab 2. Every `Card` object has a `Rank` and a `Suit`. Complete the `equals` method for `Card`.

```
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 (                                      ) {



                }











        }
}
```