1. **comparable** Implement a `compareTo` method for each of the following classes:

(a)
```
public class Shoe implements Comparable<Shoe> {
  private int size;

  @Override
  public int compareTo(Shoe other) {
    // compare shoes by size




  }
}
```

(b)
```
public class TimeOfDay implements Comparable<TimeOfDay> {
  private int hour;    // 0-23 the hour of the day
  private int minute;  // 0-59 the minute of the hour

  @Override
  public int compareTo(TimeOfDay other) {
    // compare times by hour then minute









  }
}
```

(c)
```
public class Card implements Comparable<Card> {
  private Rank rank;    // the rank of the card 2-10, J, Q, K, A
  private Suit suit;    // the suit of the card

  @Override
  public int compareTo(Card other) {
    // compare cards by their rank
    // the integer value of this.rank can be obtained
    // as this.rank.ordinal()









  }
}
```

2. **compareTo contract**

   Consider your `compareTo` method for `Shoe`:

   (a)
   ```
   Shoe x = new Shoe(8);    // size 8
   Shoe y = new Shoe(11);   // size 11
   ```

   What is the sign of:

   (a) `x.compareTo(y)`

   (b) `y.compareTo(x)`

   (c)
   ```
   Shoe x = new Shoe(7);    // size 7
   Shoe y = new Shoe(4);    // size 4
   ```

   What is the sign of:

   (a) `x.compareTo(y)`

   (b) `y.compareTo(x)`

   (c)
   ```
   Shoe x = new Shoe(7);    // size 7
   Shoe y = new Shoe(7);    // size 7
   ```

   What is the value of:

   (a) `x.compareTo(y)`

   (b) `y.compareTo(x)`

   (c) Does your `compareTo` method satisfy Part 1 of the compareTo contract?

   (d)
   ```
   Shoe x = new Shoe(8);    // size 8
   Shoe y = new Shoe(8);    // size 8
   Shoe z = new Shoe(10);   // size 10
   ```

   What is the sign of:

   (a) `x.compareTo(y)`

   (b) `x.compareTo(z)`

   (c) `y.compareTo(z)`

   (d)
   ```
   Shoe x = new Shoe(8);    // size 8
   Shoe y = new Shoe(8);    // size 8
   Shoe z = new Shoe(4);    // size 4
   ```

   What is the sign of:

   (a) `x.compareTo(y)`

   (b) `x.compareTo(z)`

   (c) `y.compareTo(z)`

   (d) Does your `compareTo` method satisfy Part 3 of the compareTo contract?

3. **Static fields**

   Modify the `Shoe` class shown below so that it keeps track of the number of shoes created. Make sure that a client is able to obtain the number of shoes created.

```
public class Shoe {
  private int size;



  public Shoe() {
    this.size = 8;


  }

  public Shoe(int size) {



  }

  public Shoe(Shoe other) {



  }







}
```

4. **UML class diagrams**

   Draw the UML class diagram for your `Shoe` class from Question 3.

5. **Multiton**

   Make `Card` into a multiton class. For the key, use the string equal to `this.rank.toString() + this.suit.toString()` You need to add a private constructor that accepts a `Rank` and a `Suit`. You also need to add a public static method that accepts a `Rank` and a `Suit` and returns the appropriate `Card` instance.

```
public class Card implements Comparable<Card> {
  private Rank rank;    // the rank of the card 2-10, J, Q, K, A
  private Suit suit;    // the suit of the card

  private static final Map<String, Card> instances =
    new HashMap<String, Card>();









}
```