Use this code for the Person class:

```
public class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

    public boolean equals(Object obj) {
        Person p = (Person)obj;
        return this.name.equals(p.name);
    }
    public String toString() {
        return this.name;
    }
}
```

1. Write 2 statements that will cause a `ClassCastException` in the equals() method.

2. Write 2 statements that will cause a `NullPointerException` in equals( ).

3. Write 2 statements that show a *different way* to cause `NullPointerException` in equals().

4. Does `toString` ever throw `NullPointerException`? What if a Person's name is null?

5. How would you fix the Person constructor to prevent a Person's name being assigned a value of null? Also *document* this behavior in the constructor's Javadoc.

```
public class Person {
    private String name;

    /**
     * Initialize a new Person object with a name.
     * @param name is the Person's name, must not be null.
     *
     */
    public Person(String name) {

        this.name = name;
    }
```

6. This `compareTo` method for the Event class is invoking the same method twice. Modify the code so that it does not need to call the same method twice.

```
public class Event implements Comparable<Event> {
    private LocalDate date;  // date of the event
```

```
    private String title;    // title of the event

    /** Order events by date.  If two events have same date,
     *  then order them by title, too.
     */
    public int compareTo(Event other) {
        if (this.date.compareTo(other.date) != 0) {
           return this.date.compareTo(other.date);
        }
        else return this.title.compareToIgnoreCase(other.title);
    }
```

Solution:

```
    public int compareTo(Event other) {




        return this.title.compareToIgnoreCase(other.title);
    }
```

7.1 Reading the spec for the methods of Stopwatch (below), what questions do you have? Is the specification complete?
7.2 Write the Stopwatch class.

`void start( )`	Start the Stopwatch
`void stop( )`	Stop the Stopwatch
`boolean isRunning()`	Returns true if stopwatch is running, otherwise false.
`double getElapsed( )`	Get the elapsed time. If stopwatch is running, it returns the elapsed time from when start invoked until present. If stopwatch is stopped, it returns the elapsed time between calls to start() and stop().

7.3 Using a separate class named Tasks, write a static method:

`public static String readFile1(String filename)`

that opens and reads a file one character at a time. Append the characters to a String and return the String.

7.4 Download the file http://se.cpe.ku.ac.th/doc/samples/Alice-in-Wonderland.txt.

7.5 Write a main method to a) use the Stopwatch to time how long it takes to read the file, b) print the elapsed time, c) print the length of the String returned.

7.6 Explain why it is so slow.

8. Write another method **`static String readFile2(String filename)`**. In this method, read the file one character at a time and append to a **`StringBuilder`** object. Return a String by calling toString() of the StringBuilder object.

8.1 Repeat 7.5 using readFile2.

8.2 Explain why the time used is so different.

9. Refactor the code to eliminate duplicate code, as described in class.