CSC207/B07 Software Design
Fall 2013 — Exercise 3

1 Logistics

- **Due date:** 10:00pm Thursday 14 November 2013
- **Group size:** Individual
- **Topics:** Design Patterns

For the rules and procedures for the exercises, including how to submit, please see the Exercises page of the course website.

2 What to do for this exercise

1. Write Java classes `Product`, `Shopper`, and `PriceWatchWebsite` that obey the specifications below. They must be in project `E3soln`, in package `e3soln`.
2. To submit your work, commit `E3soln`, `src`, `e3soln`, and your Java files under the existing directory `E3`. Do **not** commit the files and directories generated by Eclipse, such as `bin`, `.project`, etc.

3 Specifications for `Product`, `Shopper`, and `PriceWatchWebsite`

Your task is to implement `Product`, `Shopper`, and `PriceWatchWebsite` classes.

3.1 Class `Product`

A `Product` has a name, a price, and a store. Class `Product` is an `Observable` and has these methods:

- A constructor `Product(String, double, String)`, which takes the product's name, price, and store.
- A method `changePrice` that takes one double parameter that represents the new price of the product. If the new price is different from the current price, the observers should be notified with the appropriate message, which is one of:
  `The price of PRODUCT was PRICE at STORE. The price increased to NEW_PRICE.`
  OR
  `The price of PRODUCT was PRICE at STORE. The price decreased to NEW_PRICE.`

  where `PRODUCT`, `PRICE`, `STORE` and `NEW_PRICE` are the product, price, store, and new price of the product. The price and new price should be formatted to 2 decimal places using Java's format specifiers. Here is an example: `String priceFormatted = String.format("%.2f", price);`
- A method `toString` that returns a `String` of the form:
  `The price of PRODUCT was PRICE at STORE.`

  where `PRODUCT`, `PRICE`, and `STORE` are the product, price, and store of the product. The price should be formatted to 2 decimal places.

3.2 Class `Shopper`

A `Shopper` has a name. Class `Shopper` is an `Observer` and has these methods:

- A constructor `Shopper(String)`, which takes the shopper's name.
- An `update` method that prints a message when an object that the shopper is observing changes. The message is of the form:

  ```
  SHOPPER was notified about a price change.
     Notification: The price of PRODUCT was PRICE at STORE. The price increased to NEW_PRICE.
  ```

  OR

  ```
  SHOPPER was notified about a price change.
     Notification: The price of PRODUCT was PRICE at STORE. The price decreased to NEW_PRICE.
  ```

  where `SHOPPER`, `PRODUCT`, `PRICE`, `STORE`, and `NEW_PRICE` are the shopper's name, product's name, price (to 2 decimal places), store, and new price (to 2 decimal places) respectively. There are exactly three spaces before "`Notification:`".

3.3 Class `PriceWatchWebsite`

A `PriceWatchWebsite` has a URL. Class `PriceWatchWebsite` is an `Observer` **and** an `Observable`, and has these methods:

- A constructor `PriceWatchWebsite(String)`, which takes the website's URL.
- A method `update` that prints a message when an object that the shopper is observing changes. The message is of the form:

  ```
  You are subscribed to URL. It was notified about a price change.
     Notification: The price of PRODUCT was PRICE at STORE. The price increased to NEW_PRICE.
  ```

  OR

  ```
  You are subscribed to URL. It was notified about a price change.
     Notification: The price of PRODUCT was PRICE at STORE. The price decreased to NEW_PRICE.
  ```

  where `URL`, `PRODUCT`, `PRICE`, `STORE`, and `NEW_PRICE` are the website's url, product's name, price (to 2 decimal places), store, and new price (to 2 decimal places) respectively. There are exactly three spaces before "`Notification:`".

  This method also notifies its observers of the change.

4 Spacing

- After each colon, there is exactly one space.
- After each period that occurs at the end of a line, there are 0 spaces.
- After each period that occurs in the middle of a line, there is exactly 1 space.
- Before "Notification", there are exactly 3 spaces.
- The strings do not contain newline characters. (Newlines are generated using `System.out.println`).

5 Marking

The mark for correctly named files that compile and run, producing the correct output, is 3 marks. If you submit files with the correct names, but they are not in the correct directory, or do not compile, or do not belong to the correct package, or do not run, or do not produce the correct output, the solution will receive 0 marks.

6 Checklist

Have you…

- tested your code on the lab computers using **Java 1.7**?
- committed the correct files in the correct directory?
- verified that your changes were committed using `svn list` and `svn status`?
- checked the pre-marking results, made any necessary changes, and re-committed if necessary?