JAVA.UTIL.OBSERVABLE.DELETEOBSERVERS METHOD

Description

The **java.util.Observable.deleteObservers** method clears the observer list. This object will no longer have any observers.

Declaration

Following is the declaration for **java.util.Observable.deleteObservers** method

```
public void deleteObservers()
```

Parameters

- **NA**

Return Value

- **NA**

Exception

- **NA**

Example

The following example shows the usage of java.util.Observable.deleteObservers method.

```
package com.tutorialspoint;

import java.util.Observable;
import java.util.Observer;

class ObservedObject extends Observable {
   private String watchedValue;

   public ObservedObject(String value) {
   watchedValue = value;
   }

   public void setValue(String value) {
   // if value has changed notify observers
   if(!watchedValue.equals(value)) {
   System.out.println("Value changed to new value: "+value);
   watchedValue = value;

   // mark as value changed
   setChanged();
   // trigger notification
   notifyObservers(value);
   }
   }
}
public class ObservableDemo implements Observer {
   public String name;
   public ObservableDemo(String name) {
   this.name = name;
   }

   public static void main(String[] args) {
   // create watched and watcher objects
   ObservedObject watched = new ObservedObject("Original Value");
```

```
    // watcher object listens to object change
    ObservableDemo watcher1 = new ObservableDemo("Watcher1");
    ObservableDemo watcher2 = new ObservableDemo("Watcher2");

    // trigger value change
    watched.setValue("Value before addObserver");
    // add observer to the watched object
    watched.addObserver(watcher1);
    watched.addObserver(watcher2);
    // trigger value change
    watched.setValue("Value after addObserver");
    // delete all observers
    watched.deleteObservers();
    // trigger value change
    watched.setValue("Value after deleteObservers");
    }

    public void update(Observable obj, Object arg) {
    System.out.println(name+" called with Arguments: "+arg);
    }
}
```

Let us compile and run the above program, this will produce the following result:

```
Value changed to new value: Value before addObserver
Value changed to new value: Value after addObserver
Watcher2 called with Arguments: Value after addObserver
Watcher1 called with Arguments: Value after addObserver
Value changed to new value: Value after deleteObservers
```