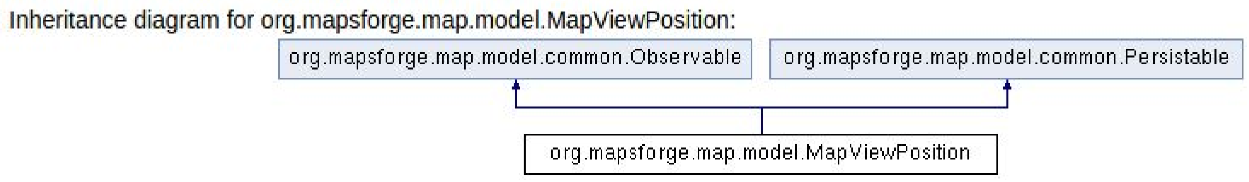
Clayton Turner
CSIS 603 - Design Patterns

Observer Pattern

Observer: MapViewPositionObserver
Code: http://code.google.com/r/ludwigbrinckmann-mapsforge/source/browse/Applications/Android/Samples/src/org/mapsforge/applications/android/samples/MapViewPositionObserver.java?name=rewrite-pom (also attached)
Observable: MapViewPosition
Code: http://grepcode.com/file/repo1.maven.org/maven2/org.mapsforge/mapsforge-map/0.5.0/org/ mapsforge/map/model/MapViewPosition.java (not in this document as not needed)

The Observer pattern defines a one-to-many dependency between objects so that when one object changes state, all dependents are notified and updated automatically. This pattern is used when a change to one object requires changing others, but these objects are loosely coupled and vary in amounts (hence the 1..* defining in the intent). An observable object, or subject, in an abstract sense, knows its observers and provides an interface for attaching and detaching the observers. The observer, in an abstract sense, defines an updating interface for objects that should be notified of changes in a subject. The concrete implementation of the subject stores state of interest and sends a notification to its observers when its state changes, whether this be a push or pull notification. The concrete implementation of the observer maintains a reference to the concrete subject object, stores state that should stay consistent with the subject's state, and implements the abstract observer updating interface to keep its state consistent with the subject.


Inheritance diagram for org.mapsforge.map.model.MapViewPosition:
org.mapsforge.map.model.common.Observable
org.mapsforge.map.model.common.Persistable
org.mapsforge.map.model.MapViewPosition

MapViewPosition is an observable object in Mapsforge. As the code is Java, this observable object extends the Observable class, albeit a custom version located within Mapsforge: http://grepcode.com/file/repo1.maven.org/maven2/org.mapsforge/mapsforge-map/0.5.0/org/mapsforge/map/model/common/Observable.java#Observable). This piece of imported code is actually very important as it defines a default implementation for the classes extending it:

```
package org.mapsforge.map.model.common;

import java.util.List;
import java.util.concurrent.CopyOnWriteArrayList;

public class Observable {
	private static final String OBSERVER_MUST_NOT_BE_NULL = "observer must not be null";
	private final List<Observer> observers = new CopyOnWriteArrayList<Observer>();
	protected Observable() {
		// do nothing
	}

	public final void addObserver(Observer observer) {
		if (observer == null) {
			throw new IllegalArgumentException(OBSERVER_MUST_NOT_BE_NULL);
		} else if (this.observers.contains(observer)) {
			throw new IllegalArgumentException("observer is already registered: " + observer);
		}
		this.observers.add(observer);
	}

	public final void removeObserver(Observer observer) {
		if (observer == null) {
			throw new IllegalArgumentException(OBSERVER_MUST_NOT_BE_NULL);
		} else if (!this.observers.contains(observer)) {
			throw new IllegalArgumentException("observer is not registered: " + observer);
		}
		this.observers.remove(observer);
	}

	protected final void notifyObservers() {
		for (Observer observer : this.observers) {
			observer.onChange();
		}
	}
}
```

Here, you can see through the functions that the Observable class is able to add observers (through addObserver(Observer observer)), remove observers (through removeObserver(Observer observer)), maintain which objects are actually observing (through the observers variable), and notify observers (through notifyObservers()). The notification update is a pull update. Each observer object has its onChange() method called. Within the observing class itself, MapViewPositionObserver, this onChange() causes the map observers to set a new center and set a new zoom (code seen below), having the observers effectively pull the information they require out.

```
package org.mapsforge.applications.android.samples;

import org.mapsforge.map.model.MapViewPosition;
import org.mapsforge.map.model.common.Observer;

class MapViewPositionObserver implements Observer {
	final private MapViewPosition observable;
	final private MapViewPosition observer;
	MapViewPositionObserver(MapViewPosition observable, MapViewPosition observer) {
		this.observable = observable;
		this.observer = observer;
		observable.addObserver(this);
	}

	void removeObserver() {
		this.observable.removeObserver(this);
	}

	@Override
	public void onChange() {
		setCenter();
		setZoom();
	}

	protected void setCenter() {
		// need to check to avoid circular notifications
		if (!this.observable.getCenter().equals(this.observer.getCenter())) {
			this.observer.setCenter(this.observable.getCenter());
		}
	}

	protected void setZoom() {
		if (this.observable.getZoomLevel() != this.observer.getZoomLevel()) {
			this.observer.setZoomLevel(this.observable.getZoomLevel());
		}
	}
}
```

As aforementioned, MapViewPositionObserver is an observer that has to pull information out of the observable object. Above, the setCenter() and setZoom() methods are clearly updating the observer object's own specific fields. Additionally, this observer maintains which object it is observing through the constructor. This constructor also contains the code to add itself to the Observable's list of objects which is observing it.