

## Exemplo de utilização do *Design Pattern*

### *Observer*

O código-fonte que será apresentado foi desenvolvido na IDE<sup>1</sup> NetBeans 7.0.1 e é um exemplo do padrão de projeto conhecido como *Observer*. A seguir é mostrada a representação em UML<sup>2</sup> e código-fonte em Java.

### Uma representação em UML do padrão de projeto *Observer*

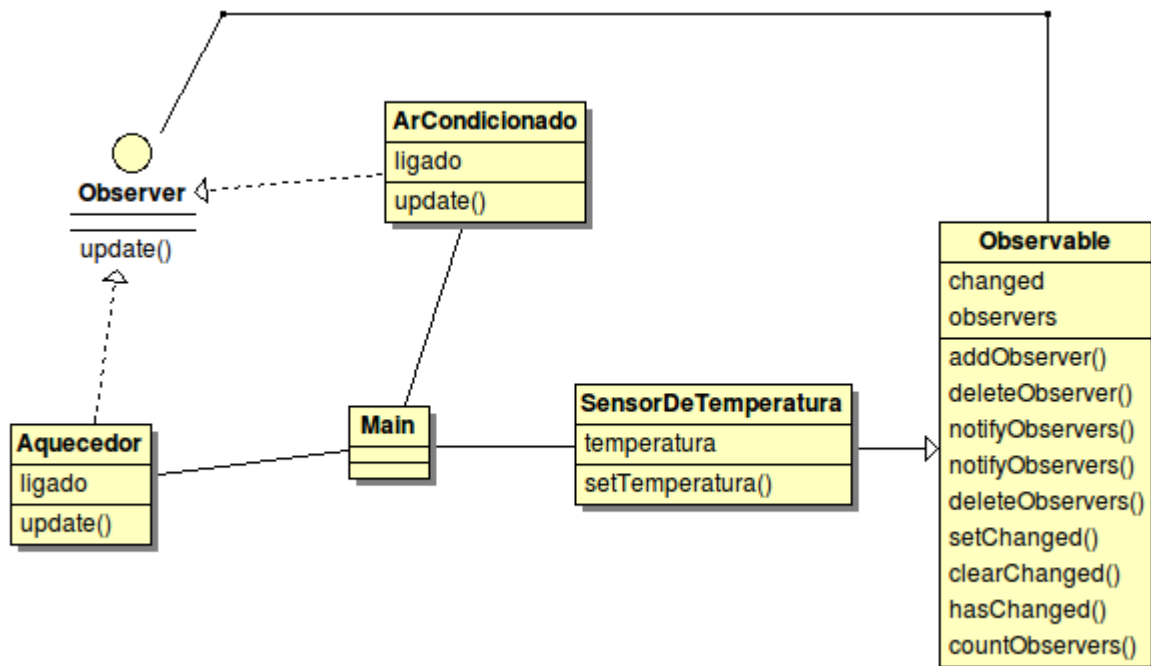


Figura 1: Modelo UML para o padrão de projeto *Observer*.

<sup>1</sup> *Integrated Development Environment*.

<sup>2</sup> *Unified Modeling Language*.

# Programação Orientada a Objetos – Java – classes *Observer* e *Observable*

Prof. Edwar Saliba Júnior – versão 1.0

## Código em Java para o modelo apresentado na Figura 1

```
1 package exemplo_padraobserver;
2
3 import java.util.Observable;
4
5 public class SensorDeTemperatura extends Observable{
6
7     private float temperatura;
8
9     public void setTemperatura(float temperatura) {
10         if (this.temperatura != temperatura) {
11             this.temperatura = temperatura;
12             setChanged();
13             notifyObservers(this.temperatura);
14         }
15     }
16 }
```

Figura 2: Classe *SensorDeTemperatura*

```
1 package exemplo_padraobserver;
2
3 import java.util.Observable;
4 import java.util.Observer;
5
6 public class ArCondicionado implements Observer {
7
8     private boolean ligado;
9
10    public ArCondicionado() {
11        this.ligado = false;
12    }
13
14    @Override
15    public void update(Observable o, Object arg) {
16        if((Integer) arg > 24 && !ligado) {
17            System.out.println("Temperatura: " + (Integer) arg
18                + " -> Ligando o ar condicionado...");
19            this.ligado = true;
20        }
21        else
22            if ((Integer) arg <= 16 && ligado) {
23                System.out.println("Temperatura: " + (Integer) arg
24                    + " -> Desligando o ar condicionado...");
25                this.ligado = false;
26            }
27    }
28 }
```

Figura 3: Classe *ArCondicionado*

## Programação Orientada a Objetos – Java – classes *Observer* e *Observable*

Prof. Edwar Saliba Júnior – versão 1.0

```
1 package exemplo_padraobserver;
2
3 import java.util.Observable;
4 import java.util.Observer;
5
6 public class Aquecedor implements Observer {
7
8     private boolean ligado;
9
10    public Aquecedor() {
11        this.ligado = false;
12    }
13
14    @Override
15    public void update(Observable o, Object arg) {
16        if((Integer) arg <= 16 && !ligado) {
17            System.out.println("Temperatura: " + (Integer) arg
18                + " -> Ligando o aquecedor...");
19            this.ligado = true;
20        }
21        else
22            if((Integer) arg > 16 && ligado) {
23                System.out.println("Temperatura: " + (Integer) arg
24                    + " -> Desligando o aquecedor...");
25                this.ligado = false;
26            }
27    }
28 }
```

Figura 4: Classe *Aquecedor*

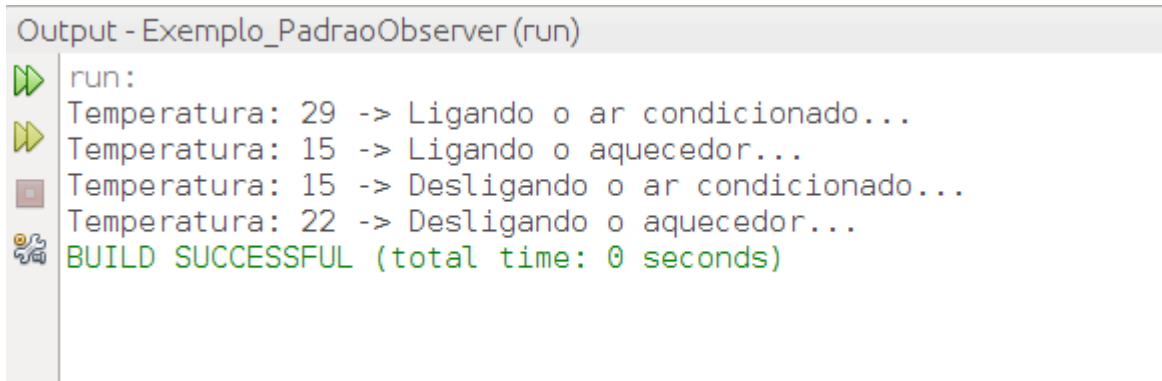
```
1 package exemplo_padraobserver;
2
3 public class Exemplo_PadraoObserver {
4
5     public static void main(String[] args) {
6
7         Aquecedor aquecedor = new Aquecedor();
8         ArCondicionado ar = new ArCondicionado();
9         SensorDeTemperatura sensor = new SensorDeTemperatura();
10
11         sensor.addObserver(ar);
12         sensor.addObserver(aquecedor);
13
14         sensor.setTemperatura(22);
15         sensor.setTemperatura(29);
16         sensor.setTemperatura(15);
17         sensor.setTemperatura(22);
18     }
19 }
```

Figura 5: Classe *Exemplo\_PadraoObserver* – método *Main*

# Programação Orientada a Objetos – Java – classes *Observer* e *Observable*

Prof. Edwar Saliba Júnior – versão 1.0

## Resultado apresentado após execução do programa



```
Output - Exemplo_PadraoObserver (run)
run:
Temperatura: 29 -> Ligando o ar condicionado...
Temperatura: 15 -> Ligando o aquecedor...
Temperatura: 15 -> Desligando o ar condicionado...
Temperatura: 22 -> Desligando o aquecedor...
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figura 6: Resultado após execução do programa.

## Bibliografia

InfoWorld – JavaWorld. **Speaking on the Observer pattern**. Disponível em: <<http://www.javaworld.com/javaworld/javaqa/2001-05/04-qa-0525-observer.html?page=2>> Acesso em: 30 dez. 2011.

Java2s. **A simple demo of Observable and Observer**. Disponível em: <<http://www.java2s.com/Code/Java/Design-Pattern/AsimpledemoofObservableandObserver.htm>> Acesso em: 30 dez. 2011.

Macedo, Alexandre. **Melhorando seu código com Design Patterns**. Disponível em: <<http://www.slideshare.net/alexmacedo/apresentacao-5925257>> Acesso em: 30 dez. 2011.

Oracle. **Classe Observable**. Documentação da linguagem Java. Disponível em: <<http://docs.oracle.com/javase/6/docs/api/java/util/Observable.html>> Acesso em: 30 dez. 2011.

Oracle. **Classe Observer**. Documentação da linguagem Java. Disponível em: <<http://docs.oracle.com/javase/6/docs/api/java/util/Observer.html>> Acesso em: 30 dez. 2011.