Instituto de Computação

Técnicas de Programação Avançada

TCC-00.174

Prof.: Anselmo Montenegro

www.ic.uff.br/~anselmo

anselmo@ic.uff.br

Conteúdo: Padrão Observer

Documento baseado no material preparado pelo Prof. Luiz André (http://www.ic.uff.br/~lapaesleme/)

Padrões de Projeto

O Padrão Observer

Como manter objetos atualizados quando algo importante ocorre?

Como fazer com que os objetos decidam se querem ser informados em tempo de execução sobre tais mudanças

Padrões de Projeto

Problema exemplo

Considere uma abstração de uma estação metereológica dada por um objeto de uma classe **WeatherData** que monitora dados de temperatura, humidade, pressão barométrica

Crie um aplicativo que forneça três elementos de exibição para: condições atuais, estatísticas metereológicas e previsão

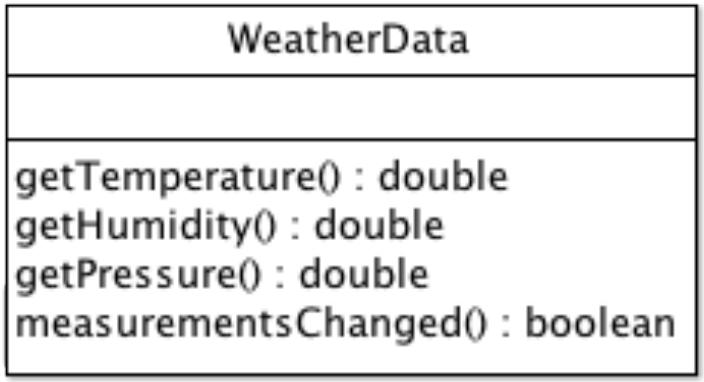
Todos os exibidores devem ser atualizados em tempo real na medida em que o objeto WeatherData adquirir medições novas

A solução deve ser expansível, isto é, outros desenvolvedores podem escrever outros exibidores de informações e os usuários podem acoplar quaisquer exibidores ao aplicativo que desejarem

Padrões de Projeto

Problema exemplo

Especificação da classe WeatherData segundo o cliente



Padrões de Projeto

Problema exemplo

```
public class WeatherData{

  // declarações de variáveis de instância
  public void measurementsChanged(){

  float temp = getTemperature();
  float humidity = getHumidity();
  float pressure = getPressure();

  currentConditionsDisplay.update(temp, humidity, pressure);
  statisticsDisplay.update(temp, humidity, pressure);
  forecastDisplay.update(temp, humidity, pressure);
  }
}
```

Padrões de Projeto

Problema exemplo

Que há de errado?

Lembrar dos conceitos e princípios da aula anterior

Padrões de Projeto

Problema exemplo

```
public class WeatherData{

  // declarações de variáveis de instância
  public void measurementsChanged(){

  float temp = getTemperature();
  float humidity = getHumidity();
  float pressure = getPressure();

  currentConditionsDisplay.update(temp, humidity, pressure);
  statisticsDisplay.update(temp, humidity, pressure);
  forecastDisplay.update(temp, humidity, pressure);
  }
}
```

Precisamos encapsular isso

Ok! Usamos uma interface (conjunto de operações comum)

A codificação voltada para implementações concretas impede a edição ou remoção dos elementos de exibição sem alteração de código

Padrões de Projeto

Solução baseada na metáfora da editora

A metáfora da editora de revistas:

Uma nova editora começa a publicar jornais

Um usuário assina a edição e recebe publicações enquanto é assinante

Quando decide não mais assinar, para de receber as publicações

Enquanto a editora continua, muitos usuários podem assinar as publicações ou cancelar as assinaturas

Padrões de Projeto

Metáfora da Editora

Observers

Subject

1

1

1

Padrões de Projeto

Metáfora da Editora

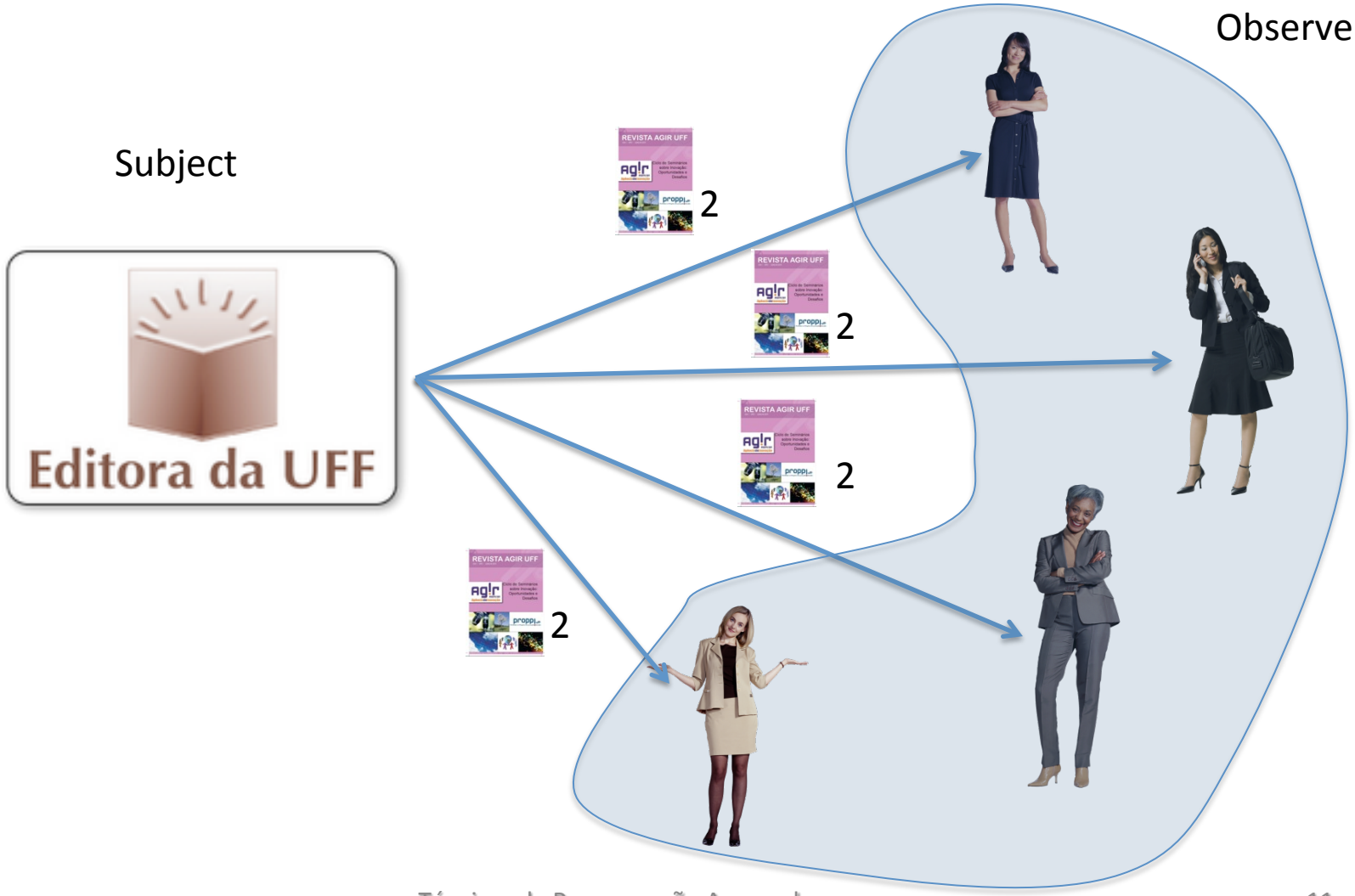
Observers

Subject

registerObserver()

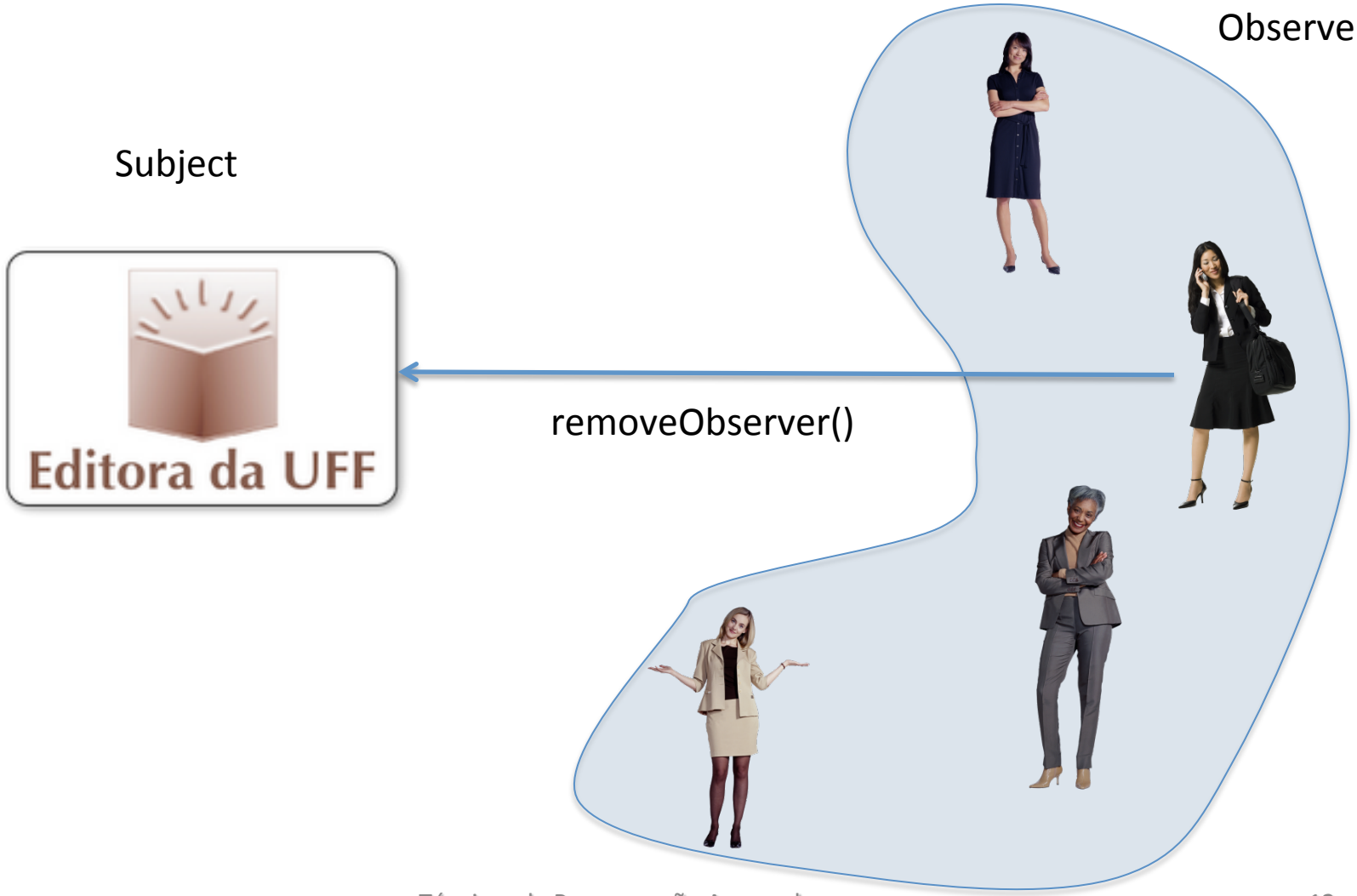
Padrões de Projeto

Benefícios de Reuso



Padrões de Projeto

Metáfora da Editora

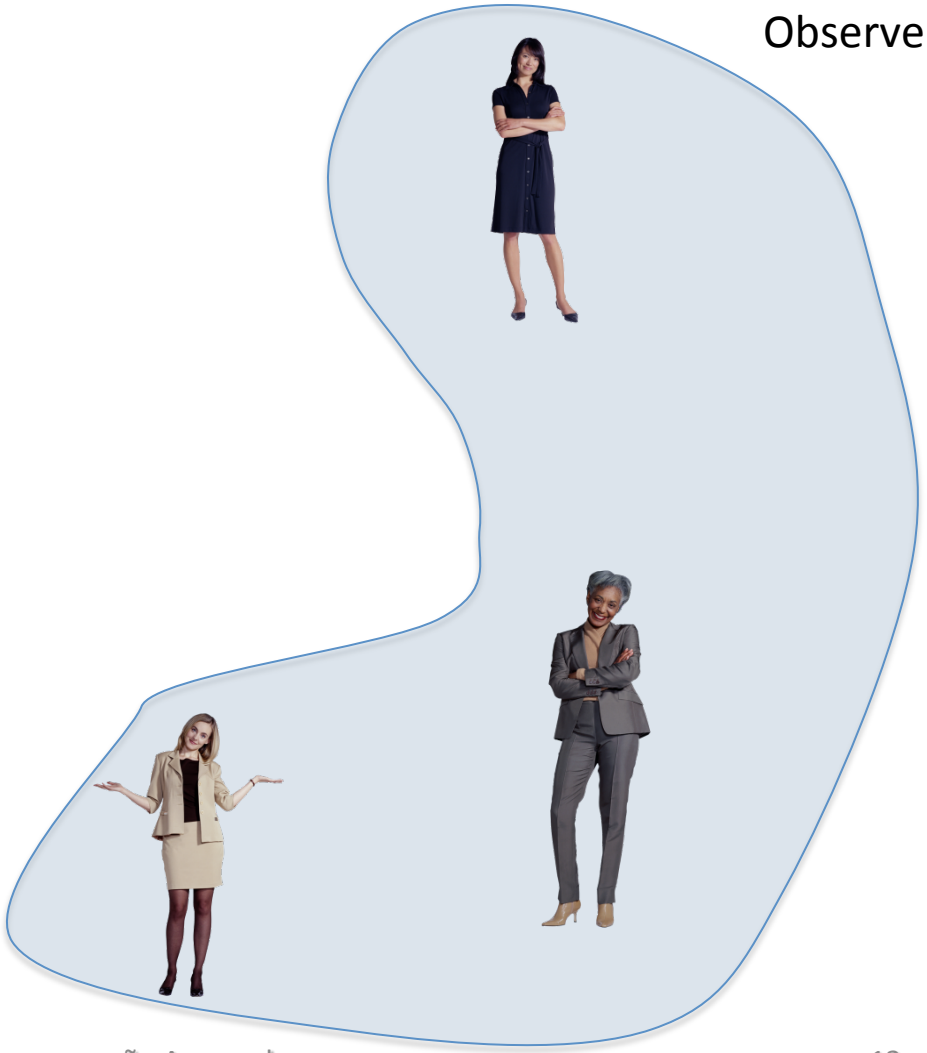


Padrões de Projeto

Metáfora da Editora

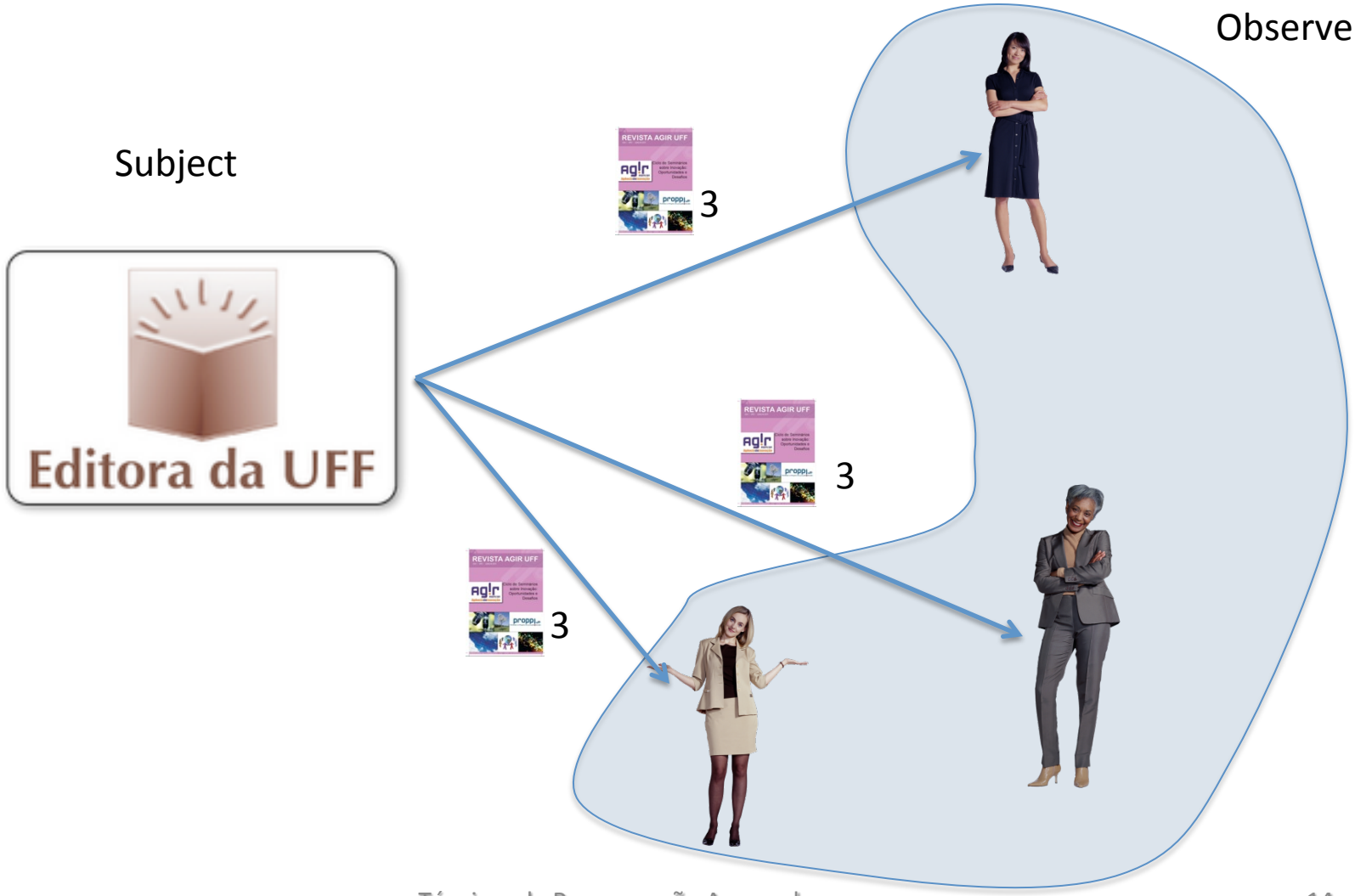
Observers

Subject



Padrões de Projeto

Benefícios de Reuso

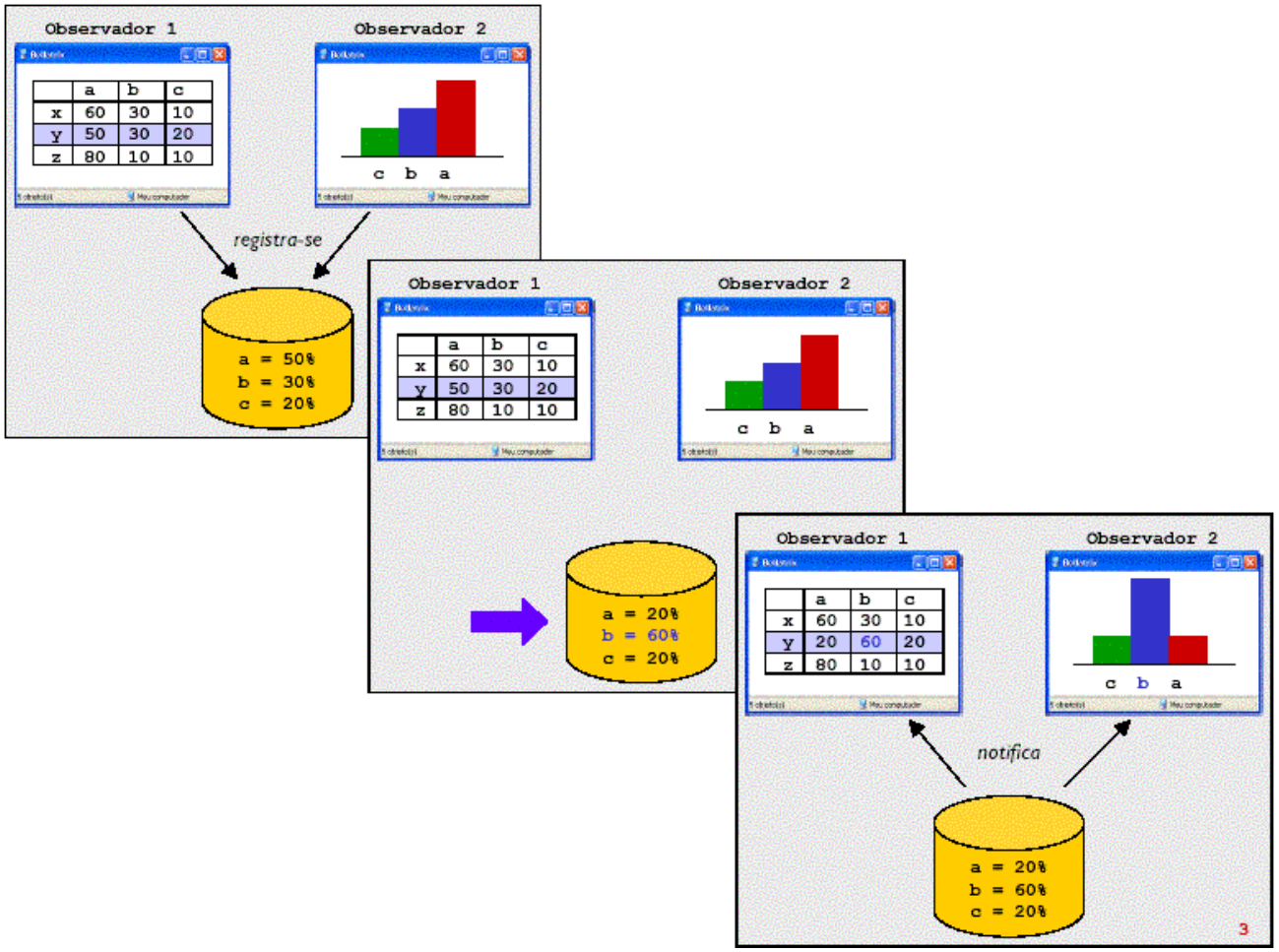


Terceiro padrão: Observer

Define uma relação de dependência um-para-muitos entre objetos de modo que quando um objeto muda de estado todos os seus dependentes são notificados e atualizados automaticamente

Padrões de Projeto

O Padrão Observer



Quarto princípio

Busque designs levemente ligados entre objetos que interagem

Os objetos que interagem devem conhecer o mínimo possível uns dos outros

Padrões de Projeto

Características do Padrão Observer

O padrão Observer fornece um design baseado em objetos onde os sujeitos e os observadores são levemente ligados

A única coisa que o sujeito sabe sob o observador é que ele implementa uma interface

Podemos adicionar novos observadores a qualquer momento

Características do Padrão Observer

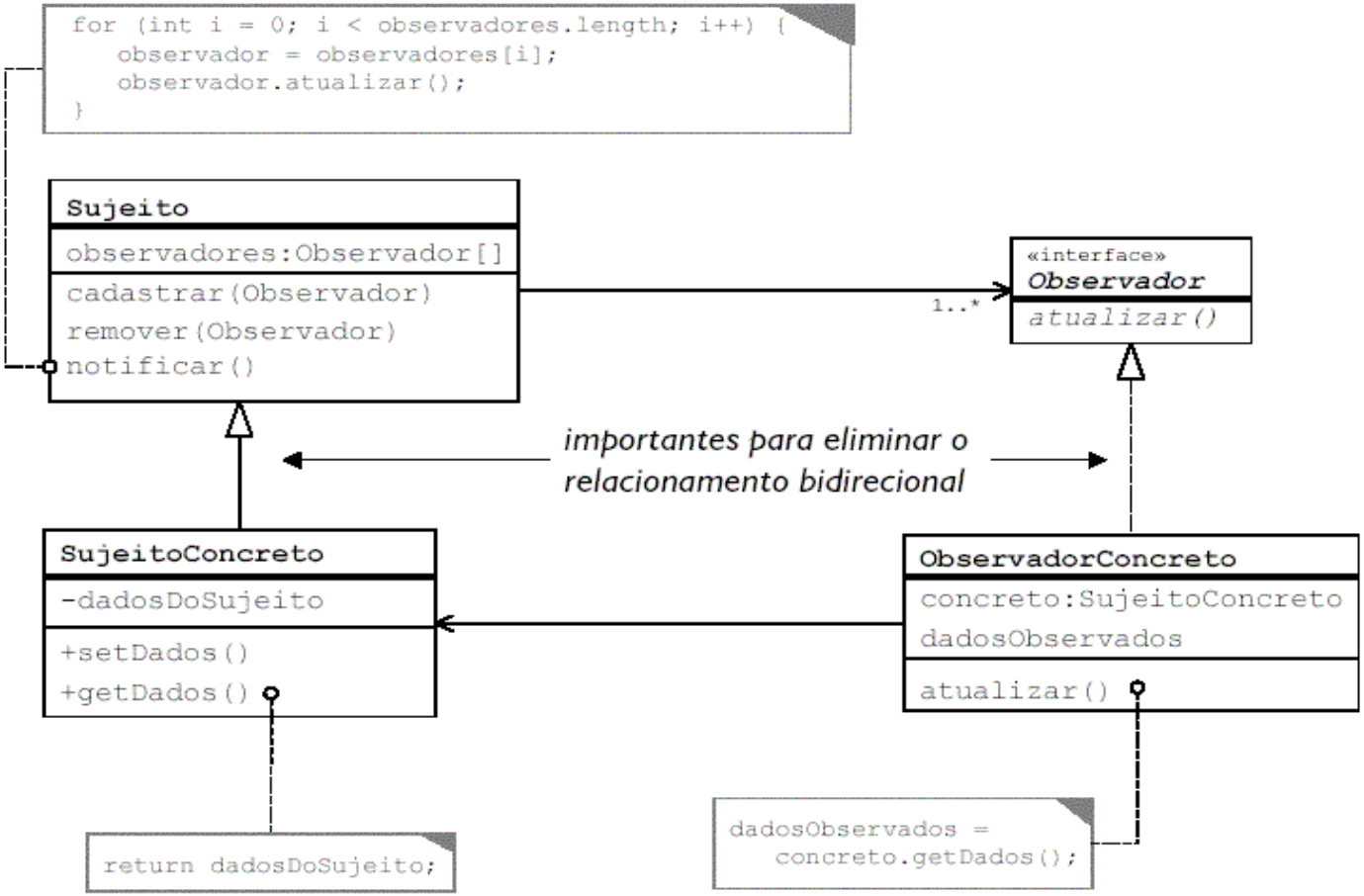
Nunca precisamos modificar o sujeito para adicionar novos tipos de observadores

Podemos reutilizar sujeitos e observadores independentemente uns dos outros

Alterações no sujeito ou num observador não irão afetar o outro

Padrões de Projeto

Padrão Observer



Padrões de Projeto

Padrão Observer

Subject

- Conhece seu Observer. Qualquer número de objetos Observer podem observar um Subject
- Provê uma interface para acoplar e desacoplar objetos Observer.

ConcreteSubject

- Guarda o estado de interesse para ConcreteObserver
- Envia uma notificação para seu Observer quando seu estado muda.

Observer

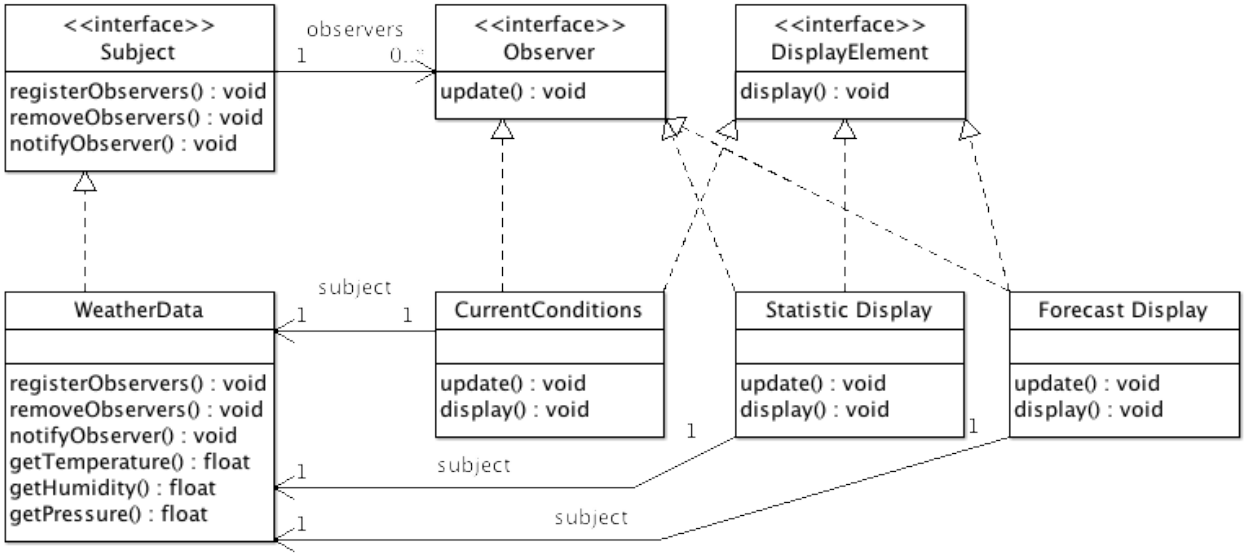
- Define uma interface de atualização para objetos que devem ser notificados sobre mudanças em um Subject.

ConcreteObserver

- Mantém uma referência para um objeto ConcreteSubject
- Guarda o estado que deve ficar consistente com o de Subject
- Implementa o Observer atualizando a interface para manter seu estado consistente com o de Subject.

Padrões de Projeto

Solução do problema exemplo baseada no Padrão Observer

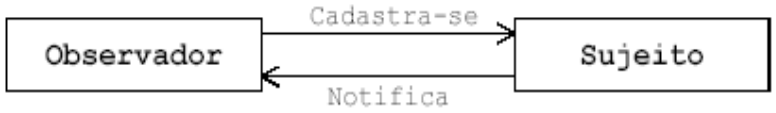


Padrões de Projeto

Propósito do Padrão Observer

Dependência de um-para-muitos entre objetos

Quando um objeto muda de estado, todos seus dependentes são notificados e atualizados automaticamente



Padrões de Projeto

Aplicabilidade do Padrão Observer

Abstração tem dois aspectos, um dependente do outro

Encapsular estes aspectos em objetos separados permite variação e reuso independentemente

Mudança em um objeto requer alterar outros

Não se sabe quantos objetos precisam ser alterados

Objeto capaz de notificar outros objetos sem presumir quem são esses objetos outros

Padrões de Projeto

Um exemplo de codificação do Padrão Observer

```
public class ConcreteObserver
             implements Observer {

  public void update(Observable o) {
    ObservableData data = (ObservableData) o;
    data.getData();
  }
}
```

```
public class ObservableData
             extends Observable {
  private Object myData;

  public void setData(Object myData) {
    this.myData = myData;
    notify();
  }

  public Object getData() {
    return myData();
  }
}
```

```
public class Observable {
  Observer observers = new ArrayList();

  public void add(Observer o) {
    observers.add(o);
  }

  public void remove(Observer o) {
    observers.remove(o);
  }

  public void notify() {
    Iterator it = observers.iterator();
    while(it.hasNext()) {
      Observer o = (Observer)it.next();
      o.update(this);
    }
  }
}
```

```
public interface Observer {
  public void update(Observable o);
}
```

Padrões de Projeto

Exercicios I

Implemente uma solução para o problema da Estação Metereológica

Estude a implementação do padrão Observer do Java

Refaça o exercício anterior usando o padrão Observer tal como implementado pelo linguagem Java. Quais as diferenças?

Padrões de Projeto

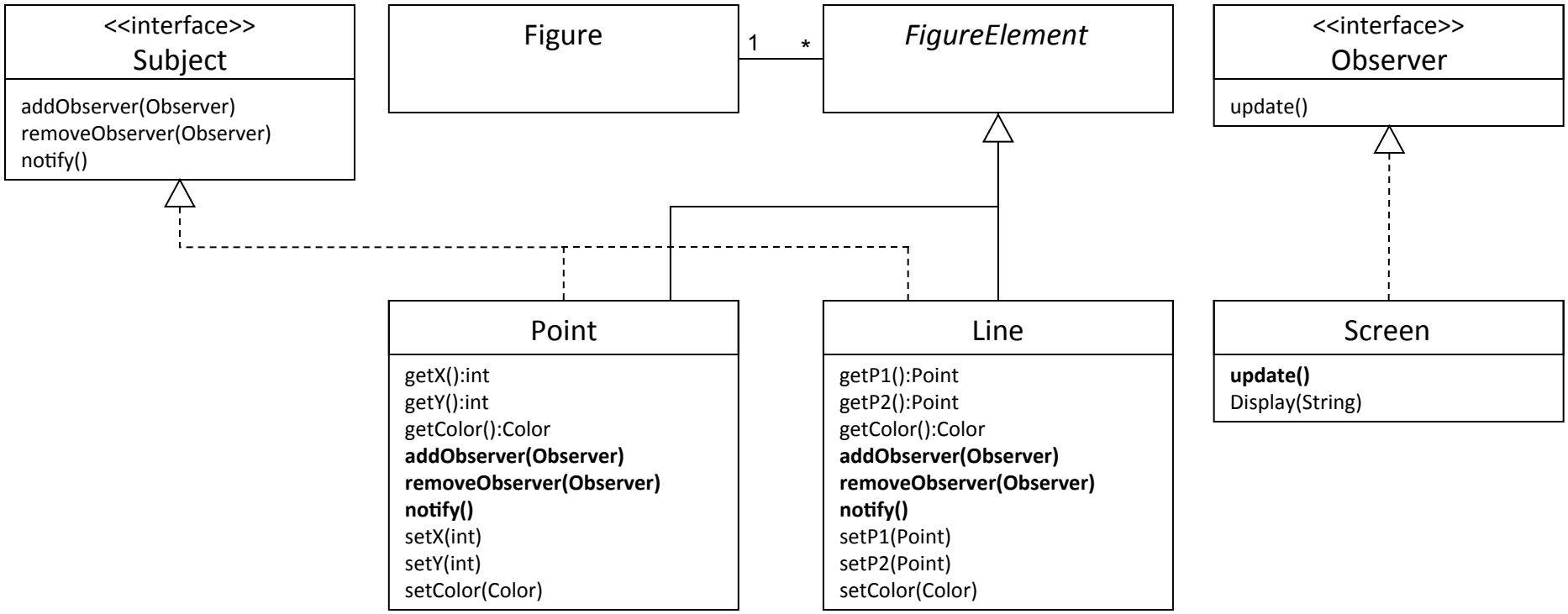
Exercicios II

Implemente um programa que exibe informações na tela sobre figuras geométricas. Suponha que existam apenas duas figuras: ponto e linha. Toda vez que um ponto ou linha tem suas propriedades modificadas, estas devem ser apresentadas na tela (inicialmente, em forma textual).

Siga o diagrama de classes do próximo slide para desenvolver sua solução

Padrões de Projeto

Exercicios



Padrões de Projeto

Bibliografia

- Use a Cabeça ! Padrões de Projetos (design Patterns) - 2ª Ed. Elisabeth Freeman e Eric Freeman. Editora: Alta Books

- Padroes de Projeto – Soluções reutilizáveis de software orientado a objetos. Erich Gamma, Richard Helm, Ralph Johnson. Editora Bookman