Análise e Projeto Orientados por Objetos

Aula 14 – Padrões GoF (Mediator e Memento)

Edirlei Soares de Lima
<edirlei@iprj.uerj.br>

Padrões GoF

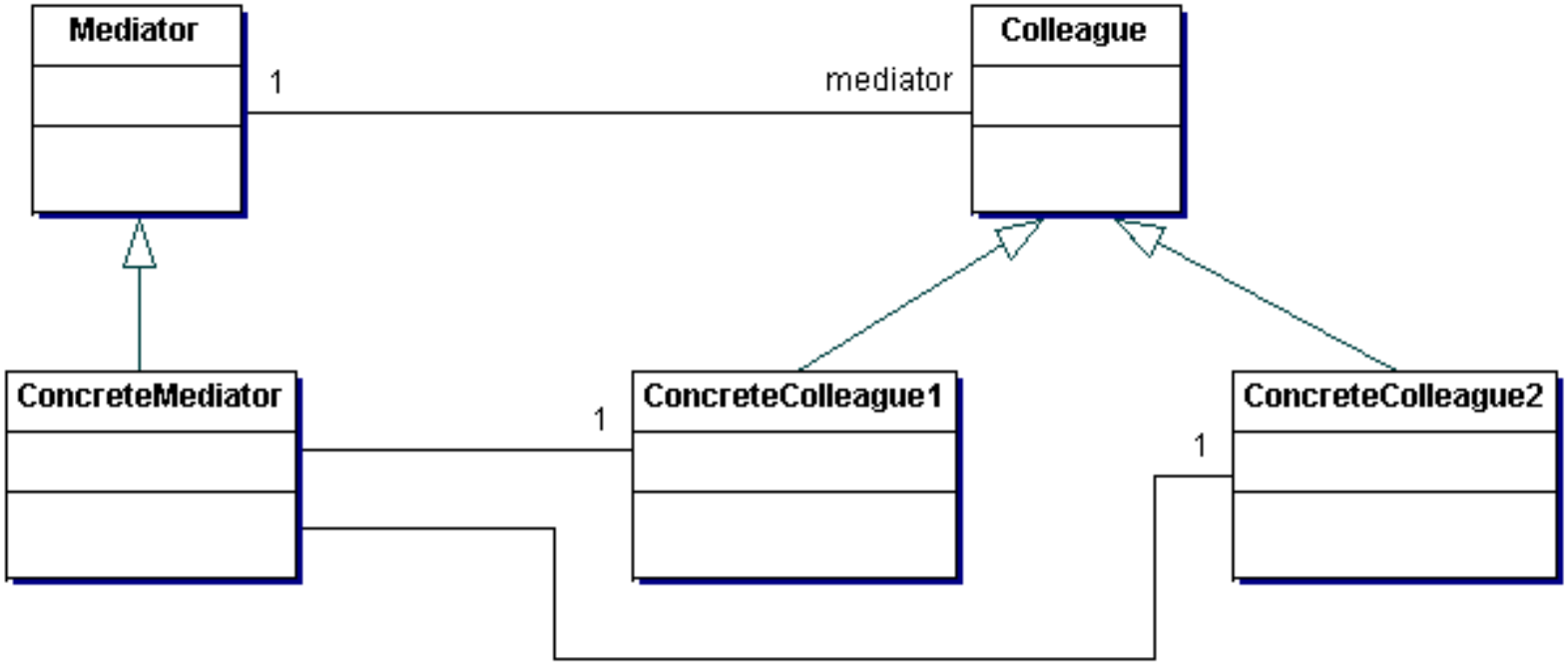
- Criação:
  - Abstract Factory
  - Builder
  - Factory Method
  - Prototype
  - Singleton
- Estruturais:
  - Adapter
  - Bridge
  - Composite
  - Decorator
  - Façade
  - Flyweight
  - Proxy
- Comportamentais:
  - Chain of Responsibility
  - Command
  - Interpreter
  - Iterator
  - **Mediator**
  - **Memento**
  - Observer
  - State
  - Strategy
  - Template Method
  - Visitor

Mediator

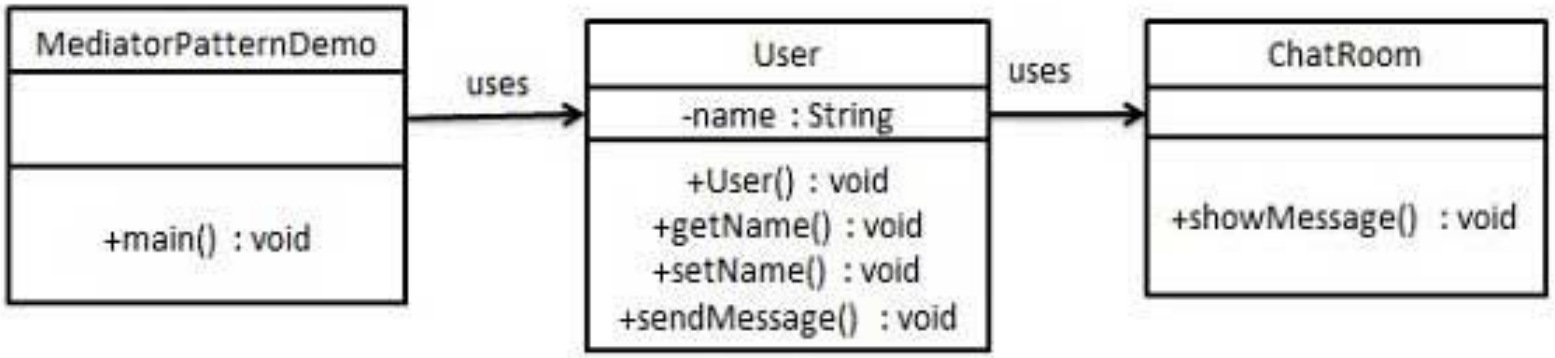
- **Intenção:** reduzir a complexidade da comunicação entre múltiplos objetos ou classes.

- **Solução:** criar uma classe mediadora para gerenciar todas as comunicações entre as diferentes classes.

Mediator


Mediator
1
mediator
Colleague
ConcreteMediator
1
ConcreteColleague1
1
ConcreteColleague2

Mediator – Exemplo



Mediator – Implementação

```
public class ChatRoom
{
   public static void showMessage(User user, String message)
   {
      System.out.println(new Date().toString() + " [" +
                         user.getName() + "] : " + message);
   }
}
```

Mediator – Implementação

```
public class User
{
   private String name;

   public String getName(){
      return name;
   }

   public void setName(String name){
      this.name = name;
   }

   public User(String name){
      this.name  = name;
   }

   public void sendMessage(String message){
      ChatRoom.showMessage(this, message);
   }
}
```

Mediator – Implementação

```
public static void main(String[] args)
{
   User robert = new User("Robert");
   User john = new User("John");

   robert.sendMessage("Hi! John!");
   john.sendMessage("Hello! Robert!");
}
```

Mediator – Consequências

- Simplifica o processo de comunicação entre múltiplos objetos ou classes;

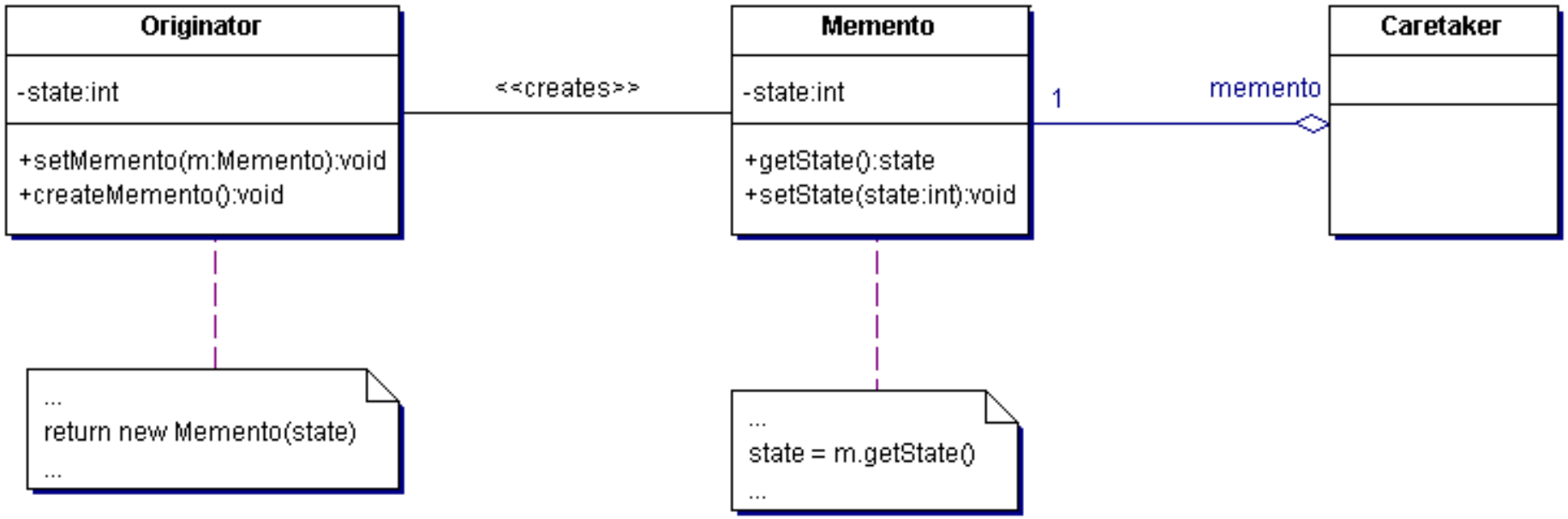
- Reduz o acoplamento entre as classes;

Memento

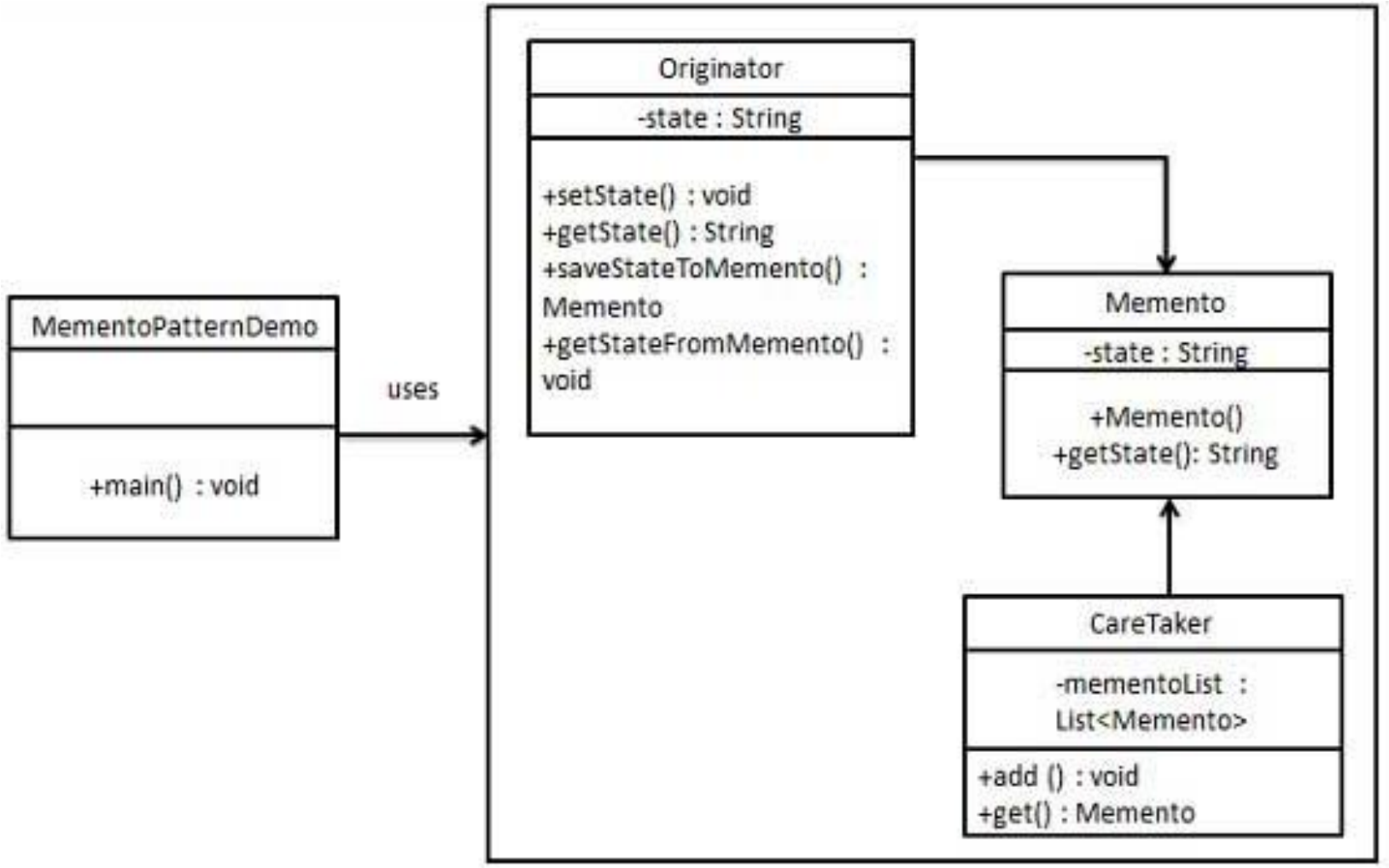
- **Intenção:** restaurar o estado de um objeto para um estado anterior.

- **Solução:** utilizar 3 classes para gerenciar os estados do objeto:
  - Memento: contem o estado de um objeto a ser restaurado;
  - Originator: cria e armazena estados em objetos Memento;
  - CareTaker: restaura o estado do objeto de Memento.

Memento


Originator
-state:int
+setMemento(m:Memento):void
+createMemento():void
<<creates>>
Memento
-state:int
+getState():state
+setState(state:int):void
1
memento
Caretaker
...
return new Memento(state)
...
...
state = m.getState()
...

Memento – Exemplo



Memento – Implementação

```
public class Memento
{
   private String state;

   public Memento(String state)
   {
      this.state = state;
   }

   public String getState()
   {
      return state;
   }
}
```

Memento – Implementação

```
public class Originator
{
   private String state;

   public void setState(String state){
      this.state = state;
   }

   public String getState(){
      return state;
   }

   public Memento saveStateToMemento(){
      return new Memento(state);
   }

   public void getStateFromMemento(Memento Memento){
      state = Memento.getState();
   }
}
```

Memento – Implementação

```
public class CareTaker
{
   private List<Memento> mementoList = new ArrayList<Memento>();

   public void add(Memento state)
   {
      mementoList.add(state);
   }

   public Memento get(int index)
   {
      return mementoList.get(index);
   }
}
```

Memento – Implementação

```
public static void main(String[] args) {

  Originator originator = new Originator();
  CareTaker careTaker = new CareTaker();

  originator.setState("State #1");
  originator.setState("State #2");
  careTaker.add(originator.saveStateToMemento());

  originator.setState("State #3");
  careTaker.add(originator.saveStateToMemento());

  originator.setState("State #4");
  System.out.println("Current State: " + originator.getState());

  originator.getStateFromMemento(careTaker.get(0));
  System.out.println("First saved State: " + originator.getState());

  originator.getStateFromMemento(careTaker.get(1));
  System.out.println("Second saved State: " + originator.getState());
}
```

Memento – Aplicabilidade

- Permite armazenar o estado interno de um objeto em um determinado momento, para que seja possível retorná-lo a este estado, caso necessário.