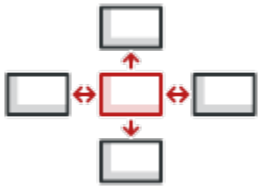
refactoring.guru

Mediator em Go / Padrões de Projeto

4–5 minutes



O **Mediator** é um padrão de projeto comportamental que reduz o acoplamento entre os componentes de um programa, fazendo-os se comunicar indiretamente, por meio de um objeto mediador especial.

O Mediator facilita a modificação, a extensão e a reutilização de componentes individuais porque eles não são mais dependentes de dezenas de outras classes.

Exemplo conceitual

Um excelente exemplo do padrão Mediator é um sistema de tráfego de estação ferroviária. Dois trens nunca se comunicam entre si pela disponibilidade da plataforma. O `stationManager` atua como um mediador e disponibiliza a plataforma para apenas um dos trens que chegam, mantendo os demais em uma fila. Um trem partindo notifica as estações, o que permite que o próximo trem na fila chegue.

train.go: Componente

```
package main

type Train interface {
    arrive()
    depart()
    permitArrival()
}
```

passengerTrain.go: Componente concreto

```
package main

import "fmt"

type PassengerTrain struct {
    mediator Mediator
}

func (g *PassengerTrain) arrive() {
    if !g.mediator.canArrive(g) {
        fmt.Println("PassengerTrain: Arrival blocked, waiting")
        return
    }
    fmt.Println("PassengerTrain: Arrived")
}

func (g *PassengerTrain) depart() {
    fmt.Println("PassengerTrain: Leaving")
    g.mediator.notifyAboutDeparture()
}
```

```
func (g *PassengerTrain) permitArrival() {
    fmt.Println("PassengerTrain: Arrival permitted, arriving")
    g.arrive()
}
```

freightTrain.go: Componente concreto

```
package main

import "fmt"

type FreightTrain struct {
    mediator Mediator
}

func (g *FreightTrain) arrive() {
    if !g.mediator.canArrive(g) {
        fmt.Println("FreightTrain: Arrival blocked, waiting")
        return
    }
    fmt.Println("FreightTrain: Arrived")
}

func (g *FreightTrain) depart() {
    fmt.Println("FreightTrain: Leaving")
    g.mediator.notifyAboutDeparture()
}

func (g *FreightTrain) permitArrival() {
```

```
    fmt.Println("FreightTrain: Arrival permitted")
    g.arrive()
}
```

mediator.go: Interface do mediador

```
package main

type Mediator interface {
    canArrive(Train) bool
    notifyAboutDeparture()
}
```

stationManager.go: Mediador concreto

```
package main

type StationManager struct {
    isPlatformFree bool
    trainQueue     []Train
}

func newStationManger() *StationManager {
    return &StationManager{
        isPlatformFree: true,
    }
}

func (s *StationManager) canArrive(t Train) bool {
    if s.isPlatformFree {
```

```
        s.isPlatformFree = false
        return true
    }
    s.trainQueue = append(s.trainQueue, t)
    return false
}

func (s *StationManager) notifyAboutDeparture() {
    if !s.isPlatformFree {
        s.isPlatformFree = true
    }
    if len(s.trainQueue) > 0 {
        firstTrainInQueue := s.trainQueue[0]
        s.trainQueue = s.trainQueue[1:]
        firstTrainInQueue.permitArrival()
    }
}
```

main.go: Código cliente

```
package main

func main() {
    stationManager := newStationManger()

    passengerTrain := &PassengerTrain{
        mediator: stationManager,
    }
    freightTrain := &FreightTrain{
        mediator: stationManager,
```

```
    }

    passengerTrain.arrive()
    freightTrain.arrive()
    passengerTrain.depart()
}
```

output.txt: Resultados da execução

```
PassengerTrain: Arrived
FreightTrain: Arrival blocked, waiting
PassengerTrain: Leaving
FreightTrain: Arrival permitted
FreightTrain: Arrived
```

Mediator em outras linguagens