Engenharia de Software II

Refatoração - Parte 2

Prof. André Hora
DCC/UFMG
2019.1

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG

Exemplo

- Refatoração por exemplo
- Programa ilustrativo no contexto de uma Locadora de DVDs
- Calcula e imprime a fatura de um cliente em uma locadora de DVDs

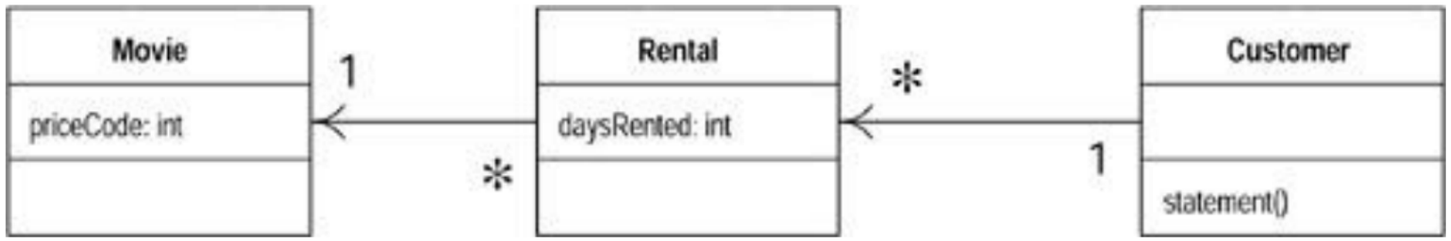
Locadora de DVDs

- Calcula e imprime a fatura de um cliente em uma locadora de DVDs
  - **Entrada**: quais filmes foram alugados e por quanto tempo
  - **Saída**: valores parciais (depende do tempo alugado e do tipo do filme), total & pontuação acumulada
- Tipos de filme: Regular, Children e New Release
- Pontuação: varia de acordo com o tipo do filme

Alteração 1: Fatura em HTML

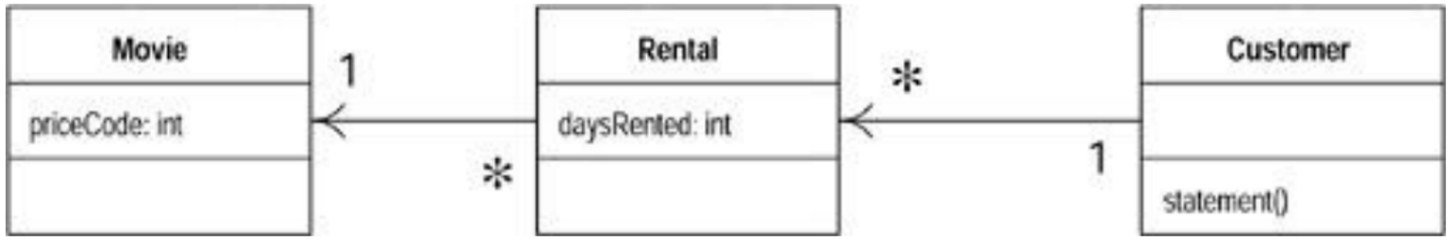
- Fatura deve ser impressa em HTML para que o sistema possa rodar na web
- Solução (ruim): copiar e colar o método `statement()` e criar `htmlStatement()`

Customer




```
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;

        String result = "Rental record for " + getName() + "\n";
        for (Rental rental : rentals) {
            double amount = 0;
            switch (rental.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                    amount += 2;
                    if (rental.getDaysRented() > 2)
                        amount += (rental.getDaysRented() - 2) * 1.5;
                    break;
                case Movie.NEW_RELEASE:
                    amount += rental.getDaysRented() * 3;
                    break;
                case Movie.CHILDREN:
                    amount += 1.5;
                    if (rental.getDaysRented() > 3)
                        amount += (rental.getDaysRented() - 3) * 1.5;
                    break;
            }

            // add frequent renter points
            frequentRenterPoints++;
            // add bonus for a two day new release rental
            if (rental.getMovie().getPriceCode() == Movie.NEW_RELEASE && rental.getDaysRented() > 1)
                frequentRenterPoints++;

            // show figures for this rental
            result += "\t" + rental.getMovie().getTitle() + "\t" + String.valueOf(amount) + "\n";

            totalAmount += amount;
        }

        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) + " frequent renter points";

        return result;
    }
}
```

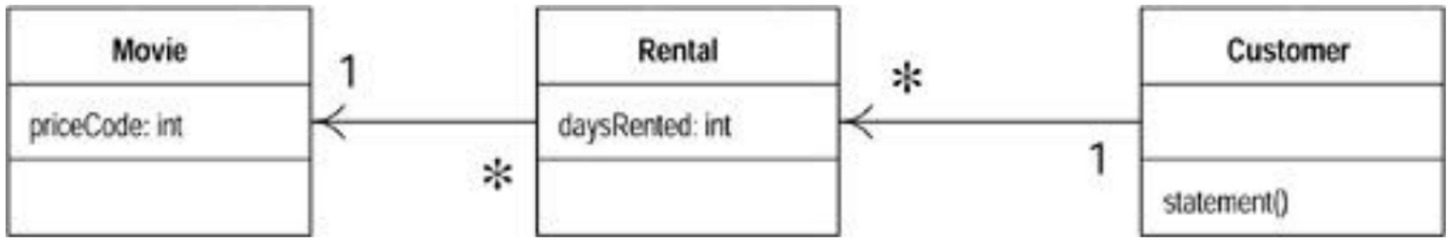
Customer




```java
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;

        String result = "Rental record for " + getName() + "\n";
        for (Rental rental : rentals) {
            double amount = 0;
            switch (rental.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                    amount += 2;
                    if (rental.getDaysRented() > 2)
                        amount += (rental.getDaysRented() - 2) * 1.5;
                    break;
                case Movie.NEW_RELEASE:
                    amount += rental.getDaysRented() * 3;
                    break;
                case Movie.CHILDREN:
                    amount += 1.5;
                    if (rental.getDaysRented() > 3)
                        amount += (rental.getDaysRented() - 3) * 1.5;
                    break;
            }

            // add frequent renter points
            frequentRenterPoints++;
            // add bonus for a two day new release rental
            if (rental.getMovie().getPriceCode() == Movie.NEW_RELEASE && rental.getDaysRented() > 1)
                frequentRenterPoints++;

            // show figures for this rental
            result += "\t" + rental.getMovie().getTitle() + "\t" + String.valueOf(amount) + "\n";

            totalAmount += amount;
        }

        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) + " frequent renter points";

        return result;
    }
}
```

Calcula preço parcial

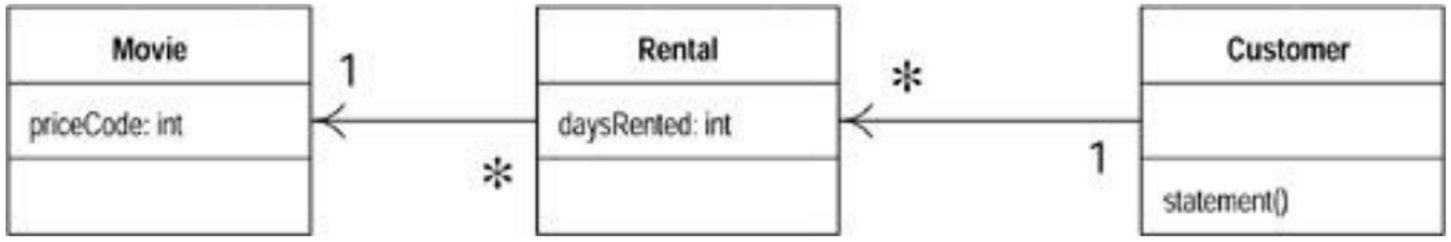
Customer




```
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;

        String result = "Rental record for " + getName() + "\n";
        for (Rental rental : rentals) {
            double amount = 0;
            switch (rental.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                    amount += 2;
                    if (rental.getDaysRented() > 2)
                        amount += (rental.getDaysRented() - 2) * 1.5;
                    break;
                case Movie.NEW_RELEASE:
                    amount += rental.getDaysRented() * 3;
                    break;
                case Movie.CHILDREN:
                    amount += 1.5;
                    if (rental.getDaysRented() > 3)
                        amount += (rental.getDaysRented() - 3) * 1.5;
                    break;
            }

            // add frequent renter points
            frequentRenterPoints++;
            // add bonus for a two day new release rental
            if (rental.getMovie().getPriceCode() == Movie.NEW_RELEASE && rental.getDaysRented() > 1)
                frequentRenterPoints++;

            // show figures for this rental
            result += "\t" + rental.getMovie().getTitle() + "\t" + String.valueOf(amount) + "\n";

            totalAmount += amount;
        }

        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) + " frequent renter points";

        return result;
    }
}
```

Calcula preço total

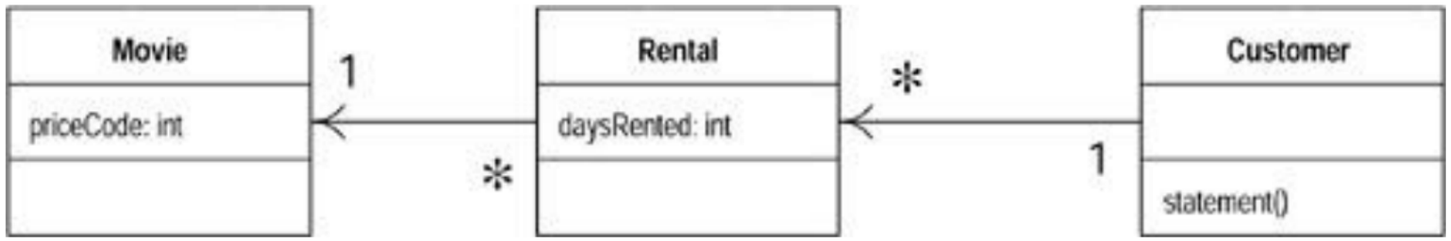
Customer




```java
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;

        String result = "Rental record for " + getName() + "\n";
        for (Rental rental : rentals) {
            double amount = 0;
            switch (rental.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                    amount += 2;
                    if (rental.getDaysRented() > 2)
                        amount += (rental.getDaysRented() - 2) * 1.5;
                    break;
                case Movie.NEW_RELEASE:
                    amount += rental.getDaysRented() * 3;
                    break;
                case Movie.CHILDREN:
                    amount += 1.5;
                    if (rental.getDaysRented() > 3)
                        amount += (rental.getDaysRented() - 3) * 1.5;
                    break;
            }

            // add frequent renter points
            frequentRenterPoints++;
            // add bonus for a two day new release rental
            if (rental.getMovie().getPriceCode() == Movie.NEW_RELEASE && rental.getDaysRented() > 1)
                frequentRenterPoints++;

            // show figures for this rental
            result += "\t" + rental.getMovie().getTitle() + "\t" + String.valueOf(amount) + "\n";

            totalAmount += amount;
        }

        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) + " frequent renter points";

        return result;
    }
}
```

Contém regras de negócio

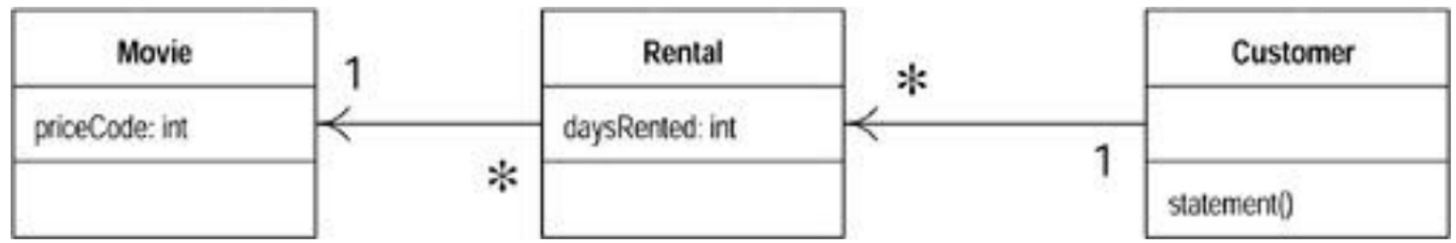
Customer




```
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;

        String result = "Rental record for " + getName() + "\n";
        for (Rental rental : rentals) {
            double amount = 0;
            switch (rental.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                    amount += 2;
                    if (rental.getDaysRented() > 2)
                        amount += (rental.getDaysRented() - 2) * 1.5;
                    break;
                case Movie.NEW_RELEASE:
                    amount += rental.getDaysRented() * 3;
                    break;
                case Movie.CHILDREN:
                    amount += 1.5;
                    if (rental.getDaysRented() > 3)
                        amount += (rental.getDaysRented() - 3) * 1.5;
                    break;
            }

            // add frequent renter points
            frequentRenterPoints++;
            // add bonus for a two day new release rental
            if (rental.getMovie().getPriceCode() == Movie.NEW_RELEASE && rental.getDaysRented() > 1)
                frequentRenterPoints++;

            // show figures for this rental
            result += "\t" + rental.getMovie().getTitle() + "\t" + String.valueOf(amount) + "\n";

            totalAmount += amount;
        }

        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) + " frequent renter points";

        return result;
    }
}
```

Calcula pontuação

Customer




```
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;

        String result = "Rental record for " + getName() + "\n";
        for (Rental rental : rentals) {
            double amount = 0;
            switch (rental.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                    amount += 2;
                    if (rental.getDaysRented() > 2)
                        amount += (rental.getDaysRented() - 2) * 1.5;
                    break;
                case Movie.NEW_RELEASE:
                    amount += rental.getDaysRented() * 3;
                    break;
                case Movie.CHILDREN:
                    amount += 1.5;
                    if (rental.getDaysRented() > 3)
                        amount += (rental.getDaysRented() - 3) * 1.5;
                    break;
            }

            // add frequent renter points
            frequentRenterPoints++;
            // add bonus for a two day new release rental
            if (rental.getMovie().getPriceCode() == Movie.NEW_RELEASE && rental.getDaysRented() > 1)
                frequentRenterPoints++;

            // show figures for this rental
            result += "\t" + rental.getMovie().getTitle() + "\t" + String.valueOf(amount) + "\n";

            totalAmount += amount;
        }

        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) + " frequent renter points";

        return result;
    }
}
```

Gera relatório

Alteração 1: Fatura em HTML

```
public String htmlStatement() {
    String result = "<h1>Rental record for <b>" + getName() + "</b></h1>\n";
    for (Rental rental : rentals)
        result += "<p>" + rental.getMovie().getTitle() + "\t" + String.valueOf(rental.getCharge()) + "</p>\n";
    result += "<p>Amount owed is <b>" + String.valueOf(getTotalCharge()) + "</b></p>\n";
    result += "<p>You earned <b>" + String.valueOf(getTotalFrequentRenterPoints()) + " frequent renter points</b></p>";
    return result;

}
```

- htmlStatement() reusa todos os métodos de cálculo
- Se as regras de cálculo mudam, só existe um lugar para alterar
- Qualquer outro formato (CSV, XML, FOO, BAR), será muito fácil de adicionar, sem duplicação de código
- Claro, o código de statement() pode ainda ser refatorado (extrair header, footer, detail)

antes

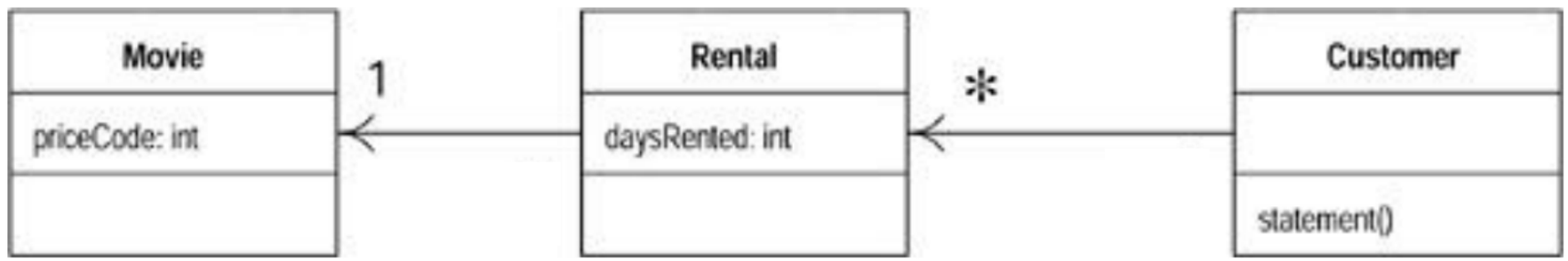
```java
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;

        String result = "Rental record for " + getName() + "\n";
        for (Rental rental : rentals) {
            double amount = 0;
            switch (rental.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                    amount += 2;
                    if (rental.getDaysRented() > 2)
                        amount += (rental.getDaysRented() - 2) * 1.5;
                    break;
                case Movie.NEW_RELEASE:
                    amount += rental.getDaysRented() * 3;
                    break;
                case Movie.CHILDREN:
                    amount += 1.5;
                    if (rental.getDaysRented() > 3)
                        amount += (rental.getDaysRented() - 3) * 1.5;
                    break;
            }

            // add frequent renter points
            frequentRenterPoints++;
            // add bonus for a two day new release rental
            if (rental.getMovie().getPriceCode() == Movie.NEW_RELEASE && rental.getDaysRented() > 1)
                frequentRenterPoints++;

            // show figures for this rental
            result += "\t" + rental.getMovie().getTitle() + "\t" + String.valueOf(amount) + "\n";

            totalAmount += amount;
        }

        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) + " frequent renter points";

        return result;
    }
}
```

depois

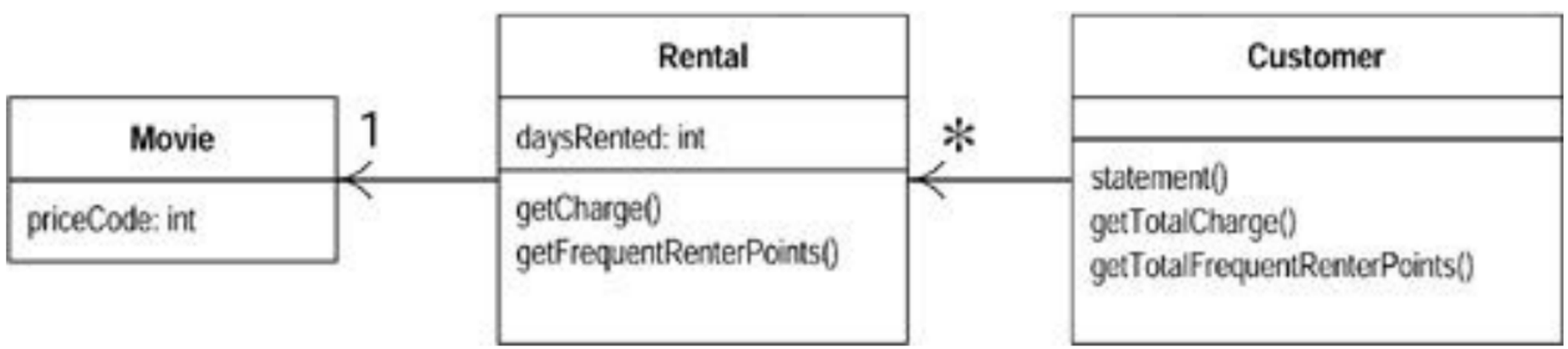
```java
public String htmlStatement() {
   String result = "<h1>Rental record for <b>" + getName() + "</b></h1>\n";
   for (Rental rental : rentals)
       result += "<p>" + rental.getMovie().getTitle() + "\t" + String.valueOf(rental.getCharge()) + "</p>\n";
   result += "<p>Amount owed is <b>" + String.valueOf(getTotalCharge()) + "</b></p>\n";
   result += "<p>You earned <b>" + String.valueOf(getTotalFrequentRenterPoints()) + " frequent renter points</b></p>";
   return result;

}
```

Diagramas de Classes

antes

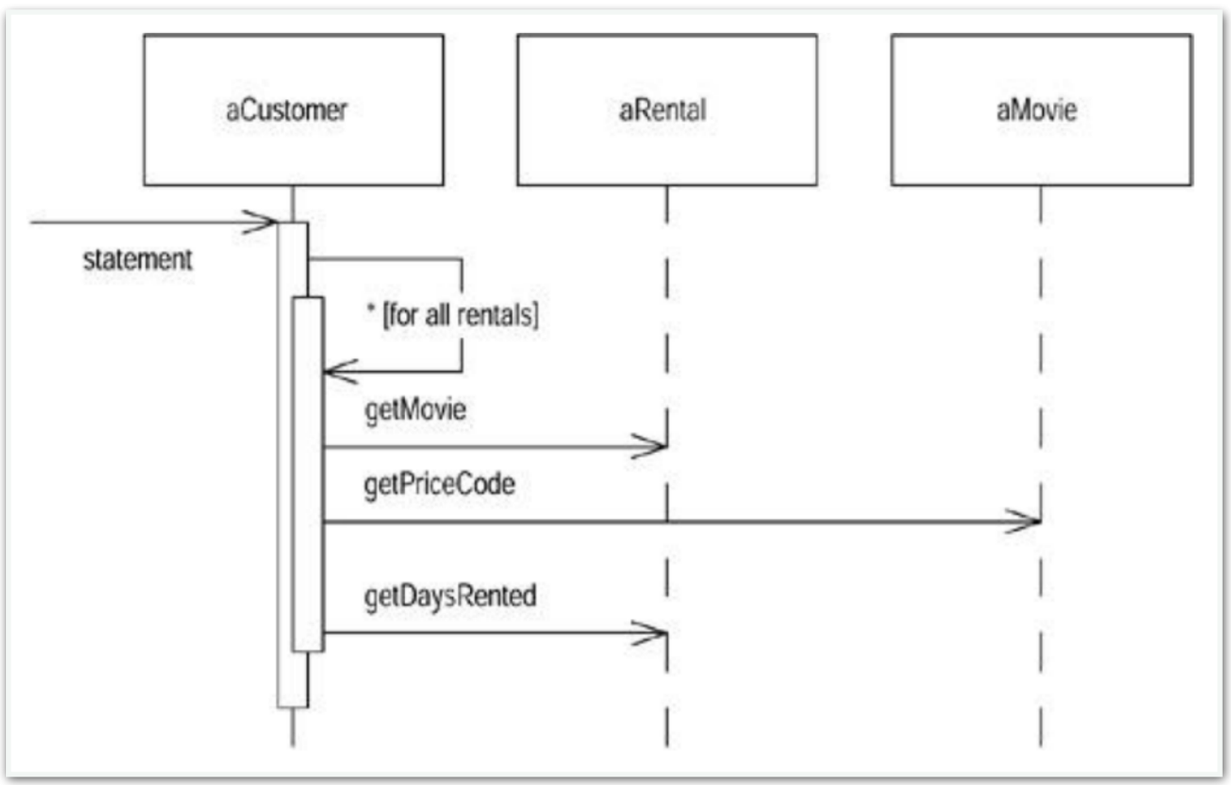


depois

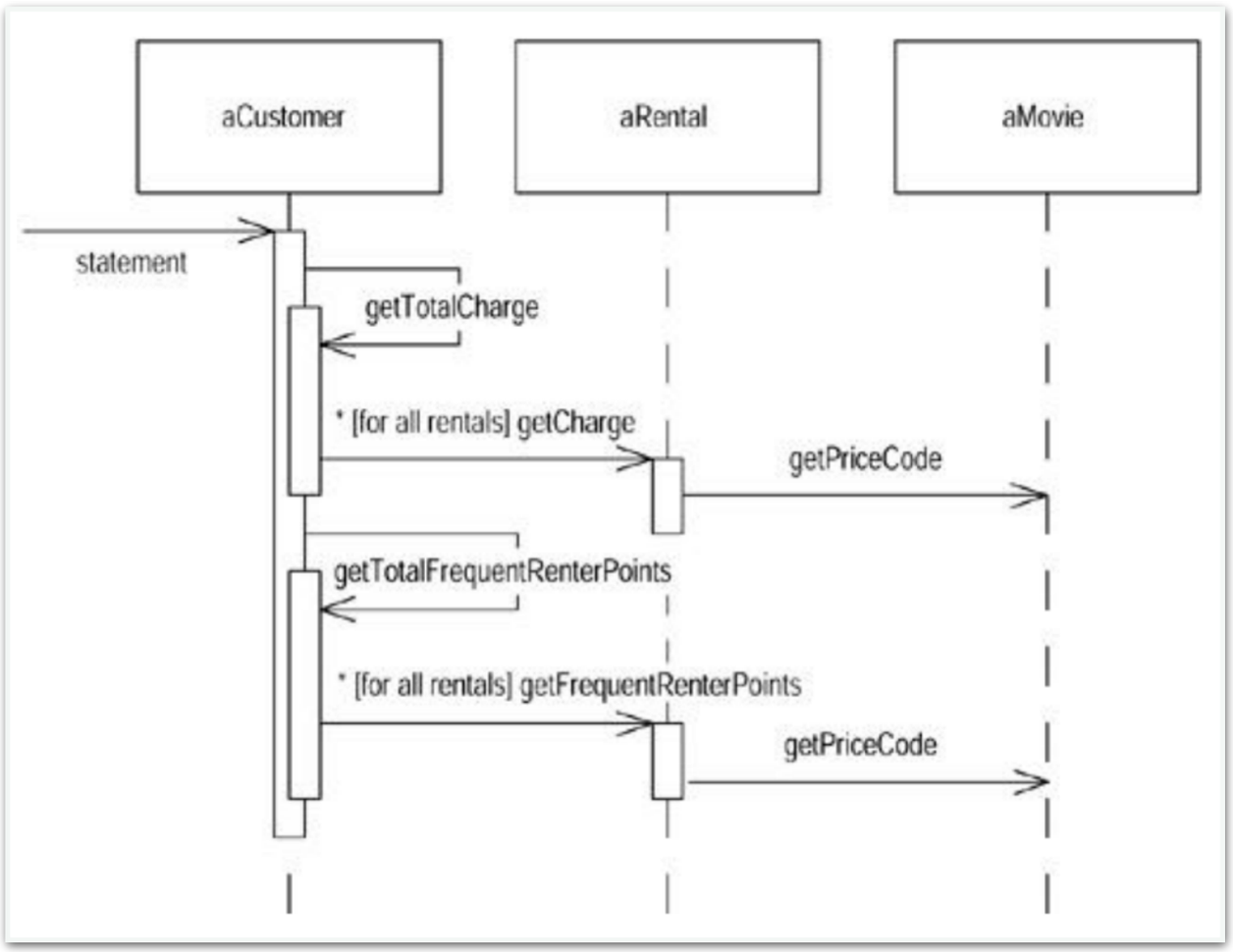


Diagramas de Sequência

antes

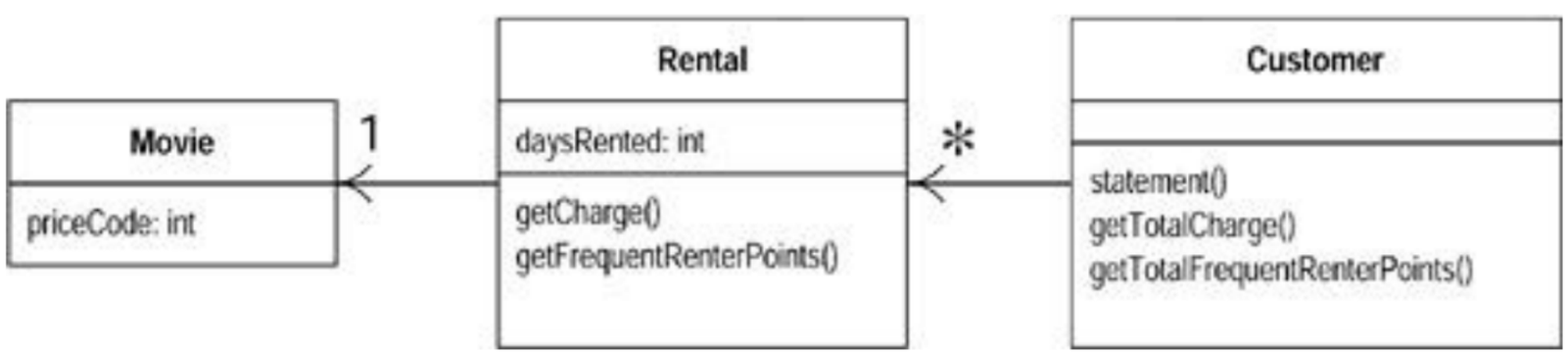


depois



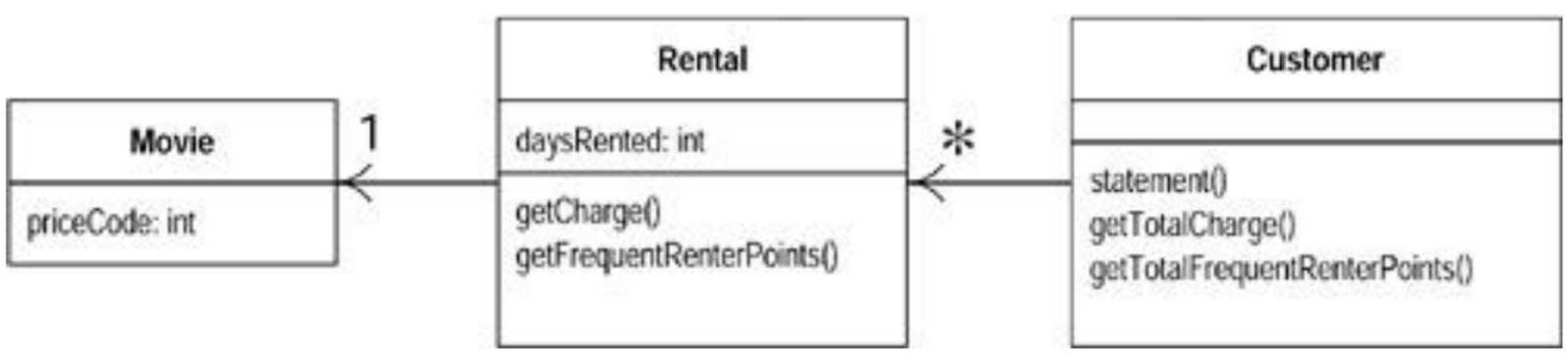
Alteração 2: Novos Tipos de Filmes

- Os tipos dos filmes vão mudar constantemente
  - Tipos hoje: Regular, Children e New Releases
  - Tipos amanhã: ?
  - Além disso, filmes podem mudar de tipo
- Importante: essa alteração afeta diretamente a forma de cobrança e o cálculo das pontuações



Exercício

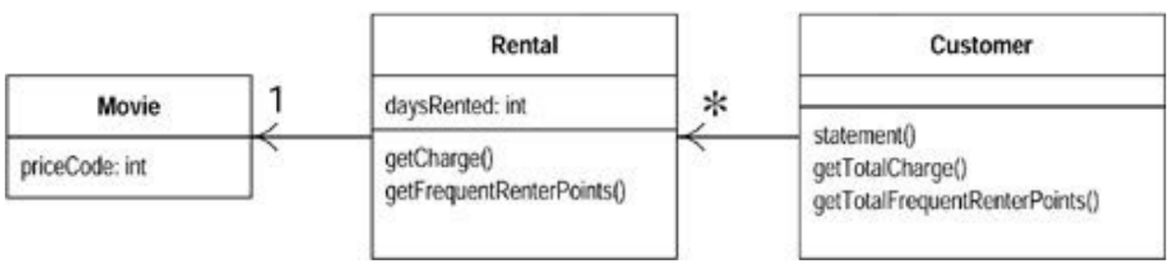
- Elaborar uma solução para adição de Novos Tipos de Filmes
- Discutir implementação
- Apresentar solução em um diagrama de classes



Movendo getCharge() de Rental para ?

Rental

```
public double getCharge() {
    double amount = 0;
    switch (getMovie().getPriceCode()) {
        case Movie.REGULAR:
            amount += 2;
            if (getDaysRented() > 2)
                amount += (getDaysRented() - 2) * 1.5;
            break;
        case Movie.NEW_RELEASE:
            amount += getDaysRented() * 3;
            break;
        case Movie.CHILDREN:
            amount += 1.5;
            if (getDaysRented() > 3)
                amount += (getDaysRented() - 3) * 1.5;
            break;
    }
    return amount;
}
```



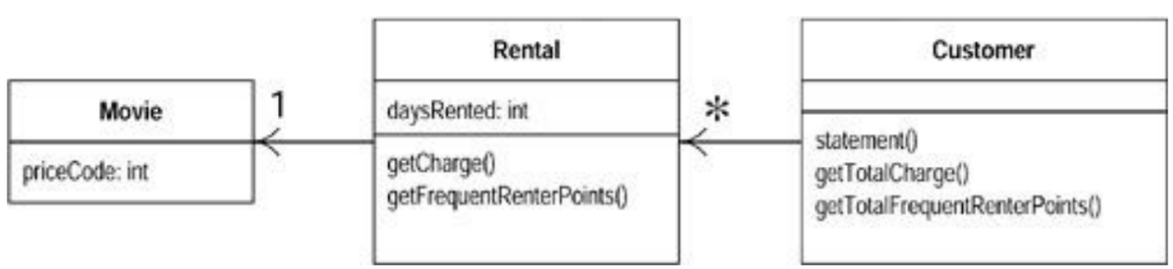
Movendo getCharge() de Rental para Movie

Rental

```
public double getCharge() {
    double amount = 0;
    switch (getMovie().getPriceCode()) {
        case Movie.REGULAR:
            amount += 2;
            if (getDaysRented() > 2)
                amount += (getDaysRented() - 2) * 1.5;
            break;
        case Movie.NEW_RELEASE:
            amount += getDaysRented() * 3;
            break;
        case Movie.CHILDREN:
            amount += 1.5;
            if (getDaysRented() > 3)
                amount += (getDaysRented() - 3) * 1.5;
            break;
    }
    return amount;
}
```

Movie

```
public double getCharge(int daysRented) {
    double amount = 0;
    switch (getPriceCode()) {
        case REGULAR:
            amount += 2;
            if (daysRented > 2)
                amount += (daysRented - 2) * 1.5;
            break;
        case NEW_RELEASE:
            amount += daysRented * 3;
            break;
        case CHILDREN:
            amount += 1.5;
            if (daysRented > 3)
                amount += (daysRented - 3) * 1.5;
            break;
    }
    return amount;
}
```



Rental

```
public double getCharge() {
    return movie.getCharge(daysRented);
}
```

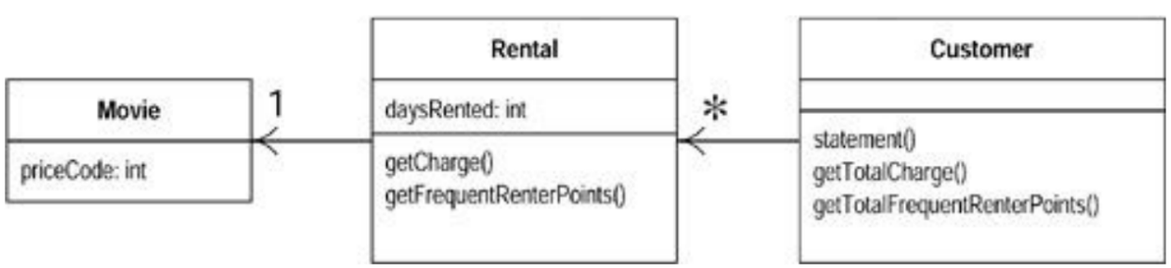
Movendo getCharge() de Rental para Movie

Rental

```
public double getCharge() {
    double amount = 0;
    switch (getMovie().getPriceCode()) {
        case Movie.REGULAR:
            amount += 2;
            if (getDaysRented() > 2)
                amount += (getDaysRented() - 2) * 1.5;
            break;
        case Movie.NEW_RELEASE:
            amount += getDaysRented() * 3;
            break;
        case Movie.CHILDREN:
            amount += 1.5;
            if (getDaysRented() > 3)
                amount += (getDaysRented() - 3) * 1.5;
            break;
    }
    return amount;
}
```

Movie

```
public double getCharge(int daysRented) {
    double amount = 0;
    switch (getPriceCode()) {
        case REGULAR:
            amount += 2;
            if (daysRented > 2)
                amount += (daysRented - 2) * 1.5;
            break;
        case NEW_RELEASE:
            amount += daysRented * 3;
            break;
        case CHILDREN:
            amount += 1.5;
            if (daysRented > 3)
                amount += (daysRented - 3) * 1.5;
            break;
    }
    return amount;
}
```



Rental

```
public double getCharge() {
    return movie.getCharge(daysRented);
}
```

Movendo getFrequentRenterPoints() de Rental para Movie

Rental

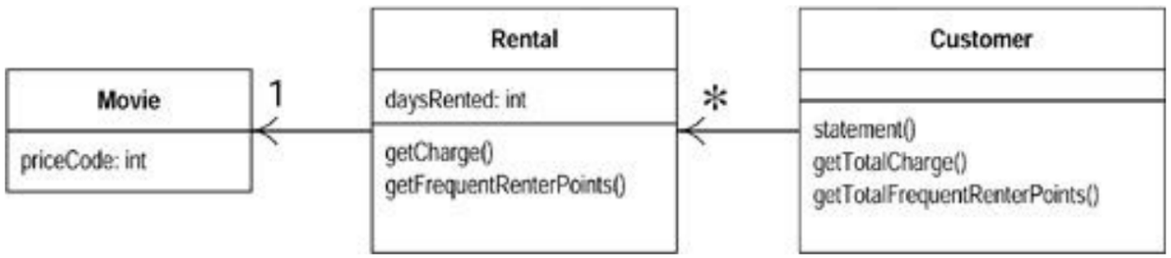
```
public int getFrequentRenterPoints() {
   if (getMovie().getPriceCode() == Movie.NEW_RELEASE && getDaysRented() > 1)
      return 2;
   return 1;
}
```

Movie

```
public int getFrequentRenterPoints(int daysRented) {
     if (priceCode == NEW_RELEASE && daysRented > 1)
          return 2;
    return 1;
}
```

Rental

```
public int getFrequentRenterPoints() {
     return movie.getFrequentRenterPoints(daysRented);
}
```



Movendo getFrequentRenterPoints() de Rental para Movie

Rental

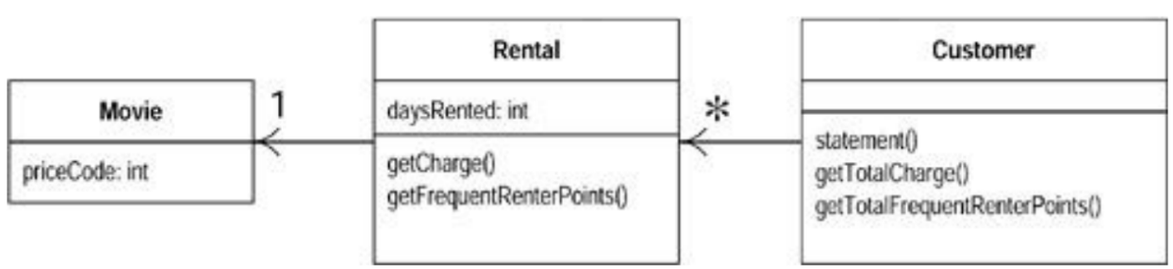
```
public int getFrequentRenterPoints() {
   if (getMovie().getPriceCode() == Movie.NEW_RELEASE && getDaysRented() > 1)
      return 2;
   return 1;
}
```

Movie

```
public int getFrequentRenterPoints(int daysRented) {
     if (priceCode == NEW_RELEASE && daysRented > 1)
          return 2;
    return 1;
}
```

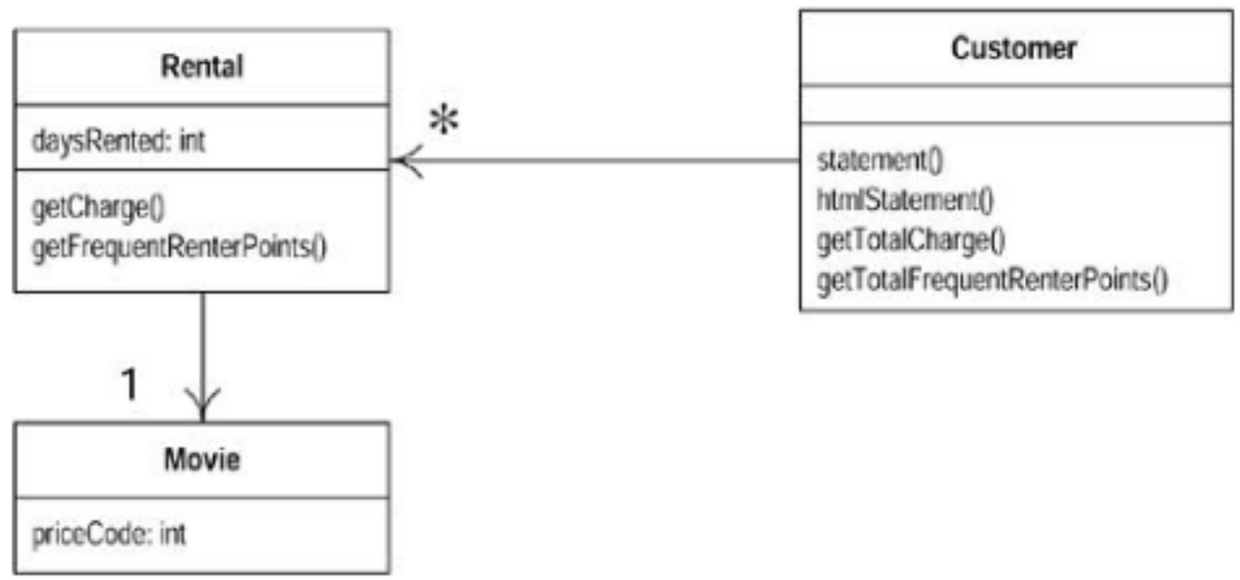
Rental

```
public int getFrequentRenterPoints() {
     return movie.getFrequentRenterPoints(daysRented);
}
```

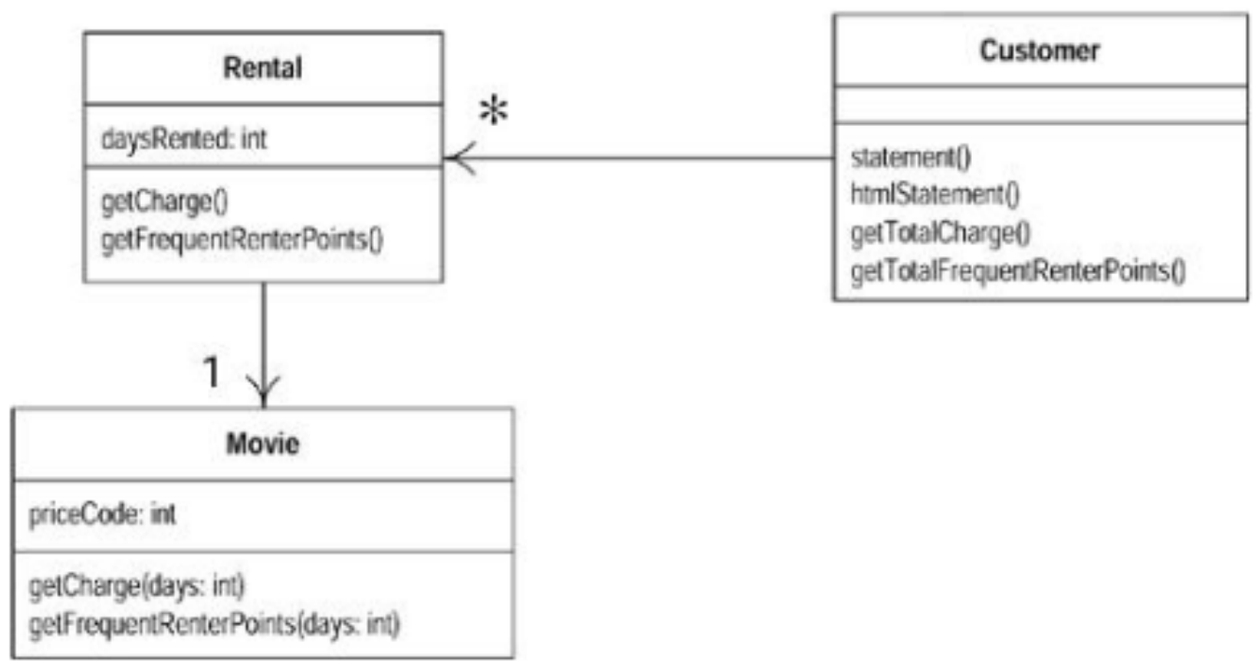


Diagramas de Classes

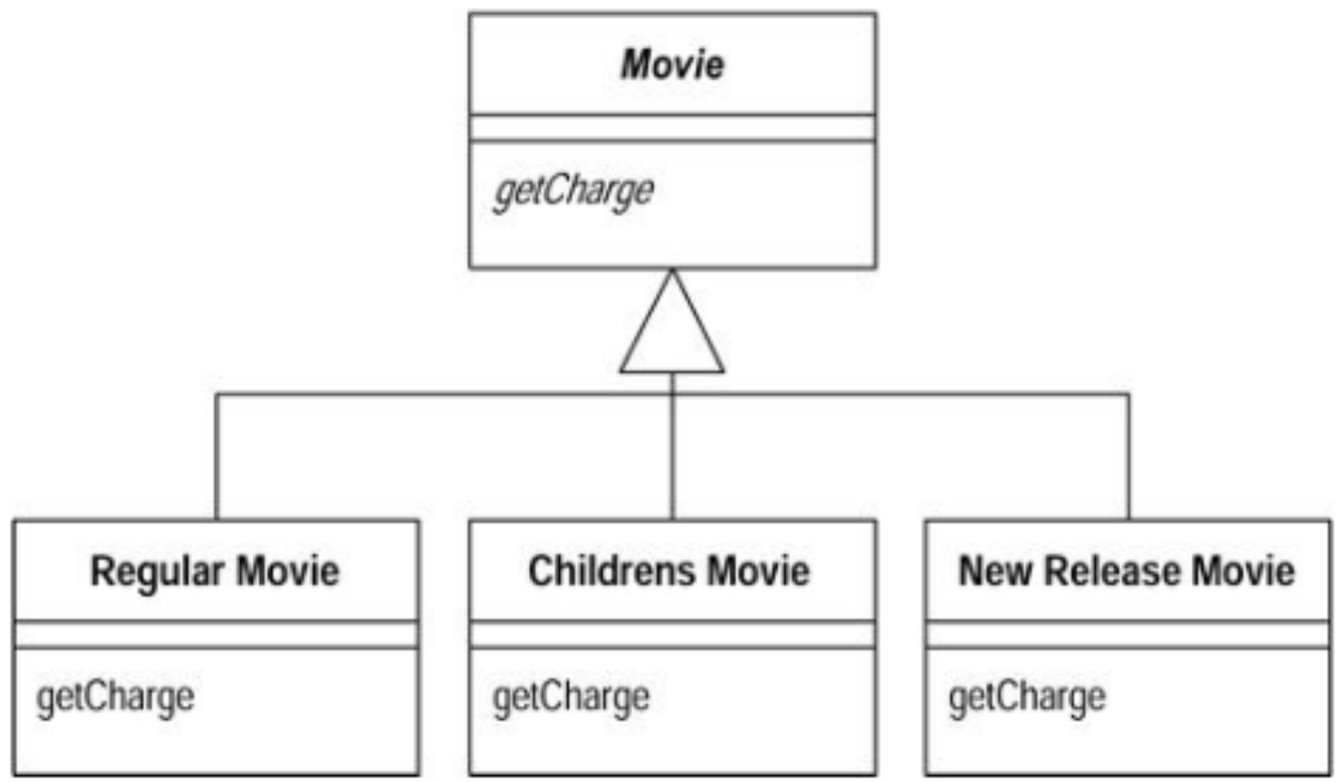
antes



depois

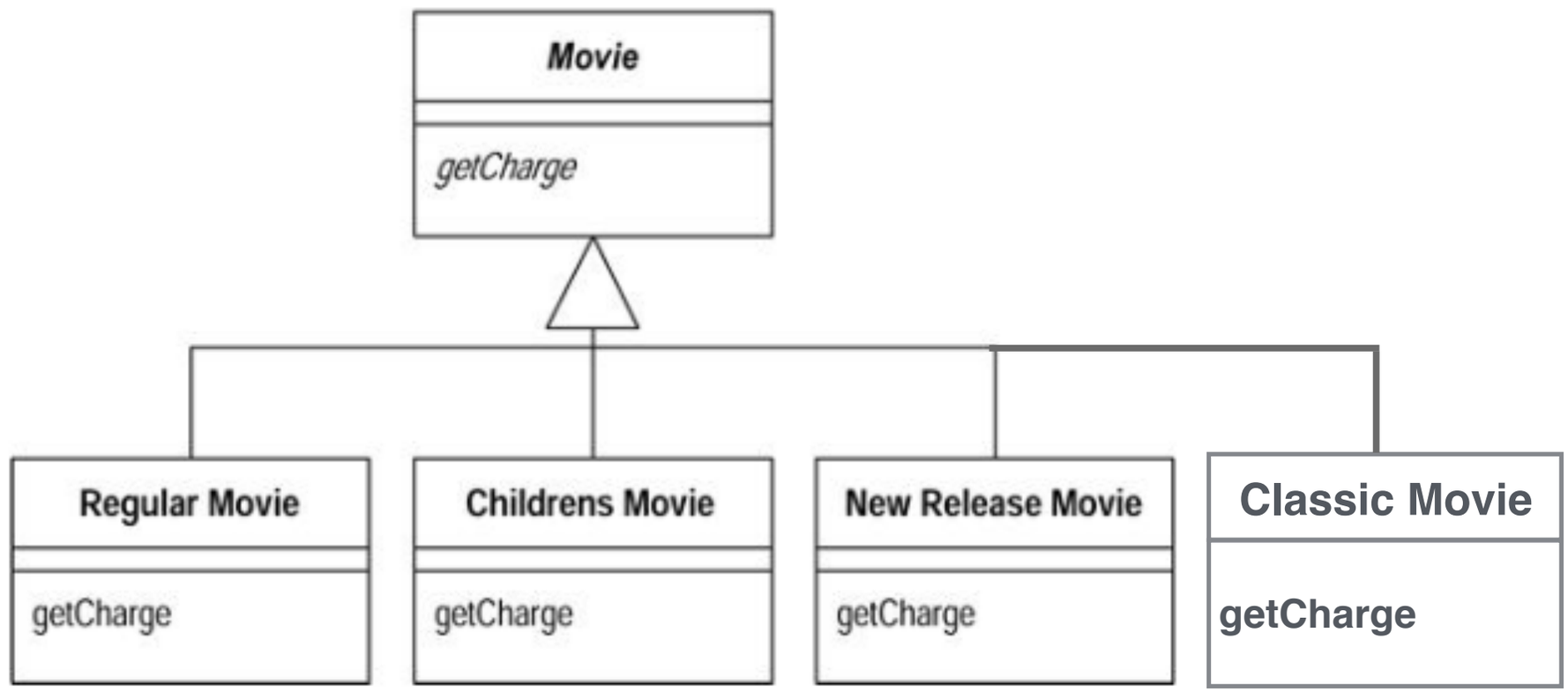


Finalmente: herança!



- Isso nos permite substituir o switch por polimorfismo
- Cada tipo de filme terá suas regras de negócio
- Novos tipos poderão ser adicionados

Finalmente: herança!

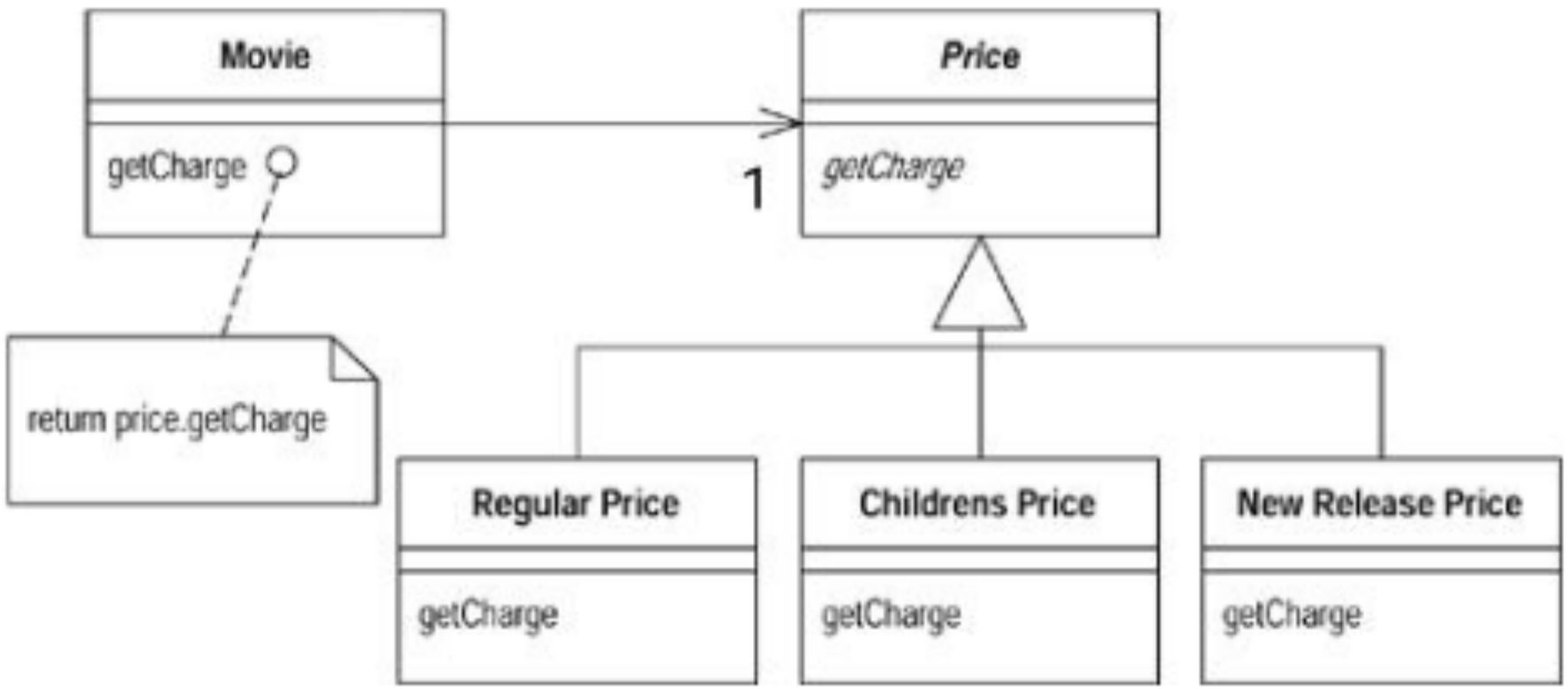


- Isso nos permite substituir o switch por polimorfismo
- Cada tipo de filme terá suas regras de negócio
- Novos tipos poderão ser adicionados

Problema

- Não funciona! (para a nossa funcionalidade)
- Um filme pode mudar de tipo
  - Ex: de New Release para Regular ao longo do tempo
- Um objeto não pode mudar sua classe ao longo do tempo
  - Logo nossa solução não funciona

Solução: State Pattern



- Permite que um objeto altere seu comportamento quando seu estado interno muda
- No nosso caso: um tipo representa um estado de filme

Refatorações

- Para introduzir o State Pattern serão necessários 3 refatorações:
  - Replace Type Code with State/Strategy Pattern
  - Move Method
  - Replace Conditional with Polymorphism

Criando novas classes

```
public abstract class Price {
    abstract int getPriceCode();
}
```

```
public class ChildrensPrice extends Price {

    public int getPriceCode() {
        return Movie.CHILDREN;
    }
}
```

```
public class RegularPrice extends Price {

    public int getPriceCode() {
        return Movie.REGULAR;
    }
}
```

```
public class NewReleasePrice extends Price {

    public int getPriceCode() {
        return Movie.NEW_RELEASE;
    }
}
```

Alterando getters e setters

```java
public class Movie {

    public static final int CHILDREN = 2;
    public static final int REGULAR = 0;
    public static final int NEW_RELEASE = 1;

    private String title;
    private int priceCode;

    public Movie(String title, int priceCode) {
        this.title = title;
        this.priceCode = priceCode;
    }

    public int getPriceCode() {
        return priceCode;
    }

    public void setPriceCode(int priceCode) {
        this.priceCode = priceCode;
    }

    public double getCharge(int daysRented) {…}
    public int getFrequentRenterPoints(int daysRented) {…}

}
```

```java
public class Movie {

    public static final int CHILDREN = 2;
    public static final int REGULAR = 0;
    public static final int NEW_RELEASE = 1;

    private String title;
    private Price price;

    public Movie(String title, int priceCode) {
        this.title = title;
        setPriceCode(priceCode);
    }

    public int getPriceCode() {
         return price.getPriceCode();
    }

    private void setPriceCode(int priceCode) {
        switch (priceCode) {
          case CHILDREN:
               price = new ChildrensPrice();
               break;
          case NEW_RELEASE:
               price = new NewReleasePrice();
               break;
          case REGULAR:
               price = new RegularPrice();
               break;
          default:
               throw new IllegalArgumentException("invalid price code");
        }
     }

    public double getCharge(int daysRented) {…}
    public int getFrequentRenterPoints(int daysRented) {…}

}
```

Alterando getters e setters

```java
public class Movie {

    public static final int CHILDREN = 2;
    public static final int REGULAR = 0;
    public static final int NEW_RELEASE = 1;

    private String title;
    private int priceCode;

    public Movie(String title, int priceCode) {
        this.title = title;
        this.priceCode = priceCode;
    }

    public int getPriceCode() {
        return priceCode;
    }

    public void setPriceCode(int priceCode) {
        this.priceCode = priceCode;
    }

    public double getCharge(int daysRented) {…}
    public int getFrequentRenterPoints(int daysRented) {…}

}
```

```java
public class Movie {

    public static final int CHILDREN = 2;
    public static final int REGULAR = 0;
    public static final int NEW_RELEASE = 1;

    private String title;
    private Price price;

    public Movie(String title, int priceCode) {
        this.title = title;
        setPriceCode(priceCode);
    }

    public int getPriceCode() {
         return price.getPriceCode();
    }

    private void setPriceCode(int priceCode) {
        switch (priceCode) {
          case CHILDREN:
               price = new ChildrensPrice();
               break;
          case NEW_RELEASE:
               price = new NewReleasePrice();
               break;
          case REGULAR:
               price = new RegularPrice();
               break;
          default:
               throw new IllegalArgumentException("invalid price code");
        }
     }

    public double getCharge(int daysRented) {…}
    public int getFrequentRenterPoints(int daysRented) {…}

}
```

Alterando getters e setters

```java
public class Movie {

    public static final int CHILDREN = 2;
    public static final int REGULAR = 0;
    public static final int NEW_RELEASE = 1;

    private String title;
    private int priceCode;

    public Movie(String title, int priceCode) {
        this.title = title;
        this.priceCode = priceCode;
    }

    public int getPriceCode() {
        return priceCode;
    }

    public void setPriceCode(int priceCode) {
        this.priceCode = priceCode;
    }

    public double getCharge(int daysRented) {…}
    public int getFrequentRenterPoints(int daysRented) {…}

}
```

```java
public class Movie {

    public static final int CHILDREN = 2;
    public static final int REGULAR = 0;
    public static final int NEW_RELEASE = 1;

    private String title;
    private Price price;

    public Movie(String title, int priceCode) {
        this.title = title;
        setPriceCode(priceCode);
    }

    public int getPriceCode() {
         return price.getPriceCode();
    }

    private void setPriceCode(int priceCode) {
        switch (priceCode) {
          case CHILDREN:
               price = new ChildrensPrice();
               break;
          case NEW_RELEASE:
               price = new NewReleasePrice();
               break;
          case REGULAR:
               price = new RegularPrice();
               break;
          default:
               throw new IllegalArgumentException("invalid price code");
        }
    }

    public double getCharge(int daysRented) {…}
    public int getFrequentRenterPoints(int daysRented) {…}

}
```

Alterando getters e setters

```java
public class Movie {

    public static final int CHILDREN = 2;
    public static final int REGULAR = 0;
    public static final int NEW_RELEASE = 1;

    private String title;
    private int priceCode;

    public Movie(String title, int priceCode) {
        this.title = title;
        this.priceCode = priceCode;
    }

    public int getPriceCode() {
        return priceCode;
    }

    public void setPriceCode(int priceCode) {
        this.priceCode = priceCode;
    }

    public double getCharge(int daysRented) {…}
    public int getFrequentRenterPoints(int daysRented) {…}

}
```

```java
public class Movie {

    public static final int CHILDREN = 2;
    public static final int REGULAR = 0;
    public static final int NEW_RELEASE = 1;

    private String title;
    private Price price;

    public Movie(String title, int priceCode) {
        this.title = title;
        setPriceCode(priceCode);
    }

    public int getPriceCode() {
        return price.getPriceCode();
    }

    private void setPriceCode(int priceCode) {
        switch (priceCode) {
          case CHILDREN:
                price = new ChildrensPrice();
                break;
          case NEW_RELEASE:
                price = new NewReleasePrice();
                break;
          case REGULAR:
                price = new RegularPrice();
                break;
          default:
                throw new IllegalArgumentException("invalid price code");
        }
    }

    public double getCharge(int daysRented) {…}
    public int getFrequentRenterPoints(int daysRented) {…}

}
```

Alterando getters e setters

```java
public class Movie {

    public static final int CHILDREN = 2;
    public static final int REGULAR = 0;
    public static final int NEW_RELEASE = 1;

    private String title;
    private int priceCode;

    public Movie(String title, int priceCode) {
        this.title = title;
        this.priceCode = priceCode;
    }

    public int getPriceCode() {
        return priceCode;
    }

    public void setPriceCode(int priceCode) {
        this.priceCode = priceCode;
    }

    public double getCharge(int daysRented) {…}
    public int getFrequentRenterPoints(int daysRented) {…}

}
```

```java
public class Movie {

    public static final int CHILDREN = 2;
    public static final int REGULAR = 0;
    public static final int NEW_RELEASE = 1;

    private String title;
    private Price price;

    public Movie(String title, int priceCode) {
        this.title = title;
        setPriceCode(priceCode);
    }

    public int getPriceCode() {
        return price.getPriceCode();
    }

    private void setPriceCode(int priceCode) {
        switch (priceCode) {
            case CHILDREN:
                price = new ChildrensPrice();
                break;
            case NEW_RELEASE:
                price = new NewReleasePrice();
                break;
            case REGULAR:
                price = new RegularPrice();
                break;
            default:
                throw new IllegalArgumentException("invalid price code");
        }
    }

    public double getCharge(int daysRented) {…}
    public int getFrequentRenterPoints(int daysRented) {…}

}
```

Movendo getCharge() de Movie para Price

Movie

```
public double getCharge(int daysRented) {
    double amount = 0;
    switch (getPriceCode()) {
        case REGULAR:
            amount += 2;
            if (daysRented > 2)
                amount += (daysRented - 2) * 1.5;
            break;
        case NEW_RELEASE:
            amount += daysRented * 3;
            break;
        case CHILDREN:
            amount += 1.5;
            if (daysRented > 3)
                amount += (daysRented - 3) * 1.5;
            break;
    }
    return amount;
}
```

Price

```
public double getCharge(int daysRented) {
    double amount = 0;
    switch (getPriceCode()) {
        case REGULAR:
            amount += 2;
            if (daysRented > 2)
                amount += (daysRented - 2) * 1.5;
            break;
        case NEW_RELEASE:
            amount += daysRented * 3;
            break;
        case CHILDREN:
            amount += 1.5;
            if (daysRented > 3)
                amount += (daysRented - 3) * 1.5;
            break;
    }
    return amount;
}
```

Movie

```
public double getCharge(int daysRented) {
    return price.getCharge(daysRented);
}
```

Replace Conditional with Polymorphism - getCharge()

Price

```
public double getCharge(int daysRented) {
    double amount = 0;
    switch (getPriceCode()) {
        case REGULAR:
            amount += 2;
            if (daysRented > 2)
                amount += (daysRented - 2) * 1.5;
            break;
        case NEW_RELEASE:
            amount += daysRented * 3;
            break;
        case CHILDREN:
            amount += 1.5;
            if (daysRented > 3)
                amount += (daysRented - 3) * 1.5;
            break;
    }
    return amount;
}
```

RegularPrice

```
public double getCharge(int daysRented) {
    double amount = 2;
    if (daysRented > 2)
        amount += (daysRented - 2) * 1.5;
    return amount;
}
```

NewReleasePrice

```
public double getCharge(int daysRented) {
    return daysRented * 3;
}
```

ChildrensPrice

```
public double getCharge(int daysRented) {
    double amount = 1.5;
    if (daysRented > 3)
        amount += (daysRented - 3) * 1.5;
    return amount;
}
```

Price

```
abstract double getCharge(int daysRented);
```

Replace Conditional with Polymorphism - getCharge()

Price

```
public double getCharge(int daysRented) {
    double amount = 0;
    switch (getPriceCode()) {
        case REGULAR:
            amount += 2;
            if (daysRented > 2)
                amount += (daysRented - 2) * 1.5;
            break;
        case NEW_RELEASE:
            amount += daysRented * 3;
            break;
        case CHILDREN:
            amount += 1.5;
            if (daysRented > 3)
                amount += (daysRented - 3) * 1.5;
            break;
    }
    return amount;
}
```

RegularPrice

```
public double getCharge(int daysRented) {
    double amount = 2;
    if (daysRented > 2)
        amount += (daysRented - 2) * 1.5;
    return amount;
}
```

NewReleasePrice

```
public double getCharge(int daysRented) {
    return daysRented * 3;
}
```

ChildrensPrice

```
public double getCharge(int daysRented) {
    double amount = 1.5;
    if (daysRented > 3)
        amount += (daysRented - 3) * 1.5;
    return amount;
}
```

Price

```
abstract double getCharge(int daysRented);
```

Replace Conditional with Polymorphism - getFrequentRenterPoints()

Movie

```
public int getFrequentRenterPoints(int daysRented) {
    if (priceCode == NEW_RELEASE && daysRented > 1)
        return 2;
    return 1;
}
```

NewReleasePrice

```
public int getFrequentRenterPoints(int daysRented) {
    if (daysRented > 1)
        return 2;
    return 1;
}
```

Price

```
public int getFrequentRenterPoints(int daysRented) {
    return 1;
}
```

Replace Conditional with Polymorphism - getFrequentRenterPoints()

Movie

```
public int getFrequentRenterPoints(int daysRented) {
    if (priceCode == NEW_RELEASE && daysRented > 1)
        return 2;
    return 1;
}
```

NewReleasePrice

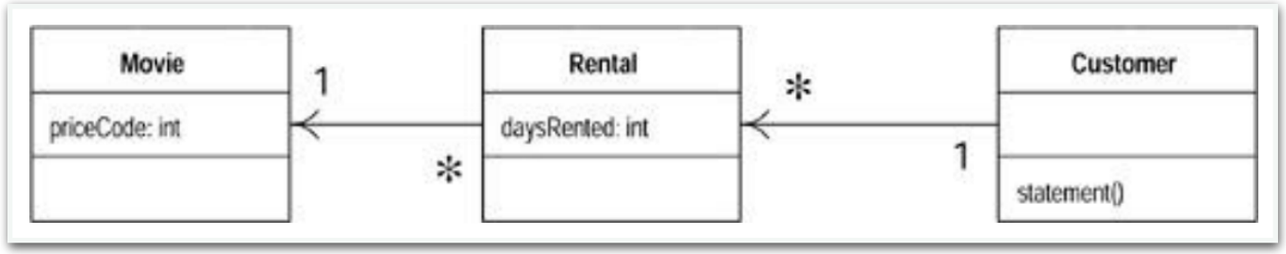
```
public int getFrequentRenterPoints(int daysRented) {
    if (daysRented > 1)
        return 2;
    return 1;
}
```

Price

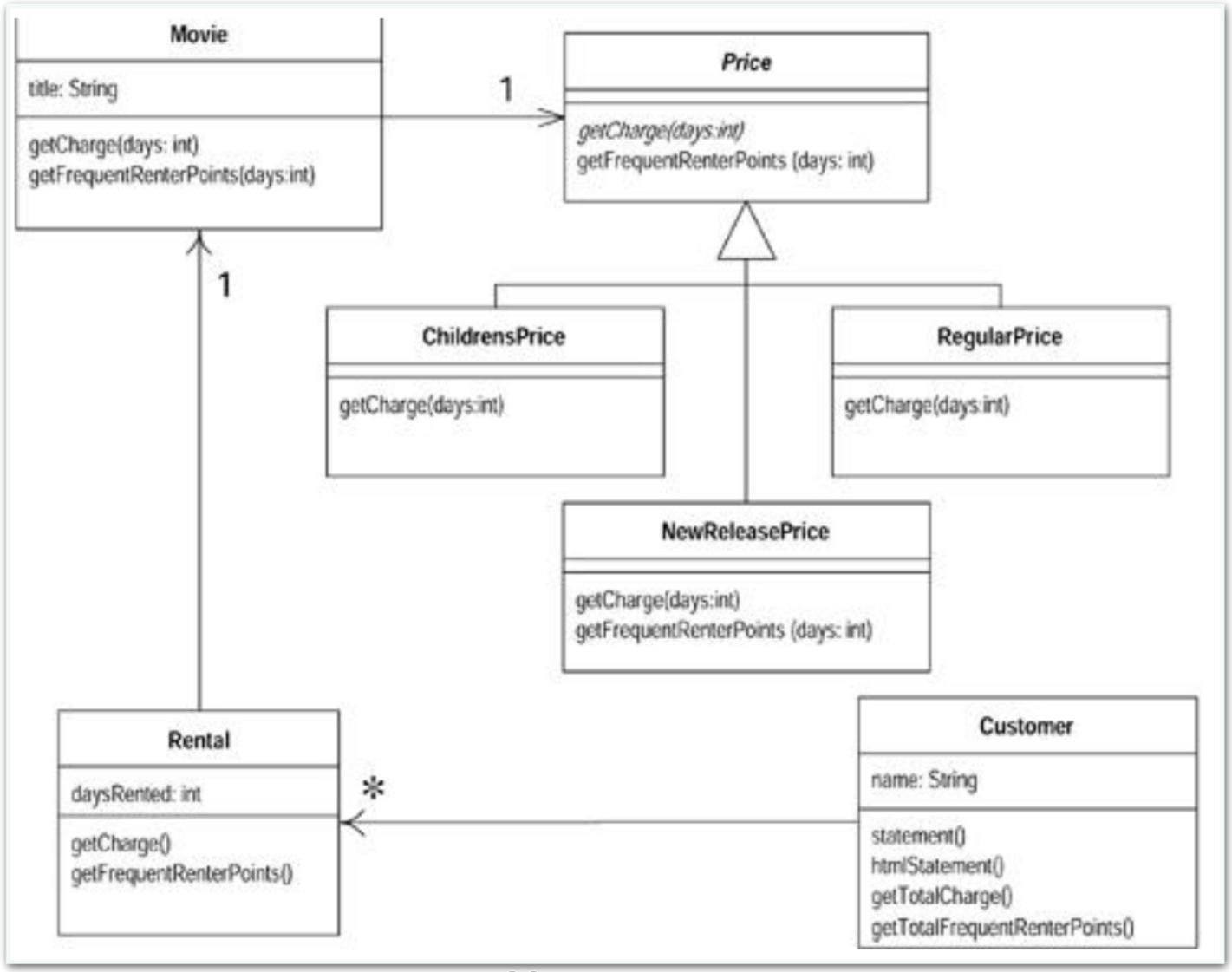
```
public int getFrequentRenterPoints(int daysRented) {
    return 1;
}
```

Diagramas de Classes

antes

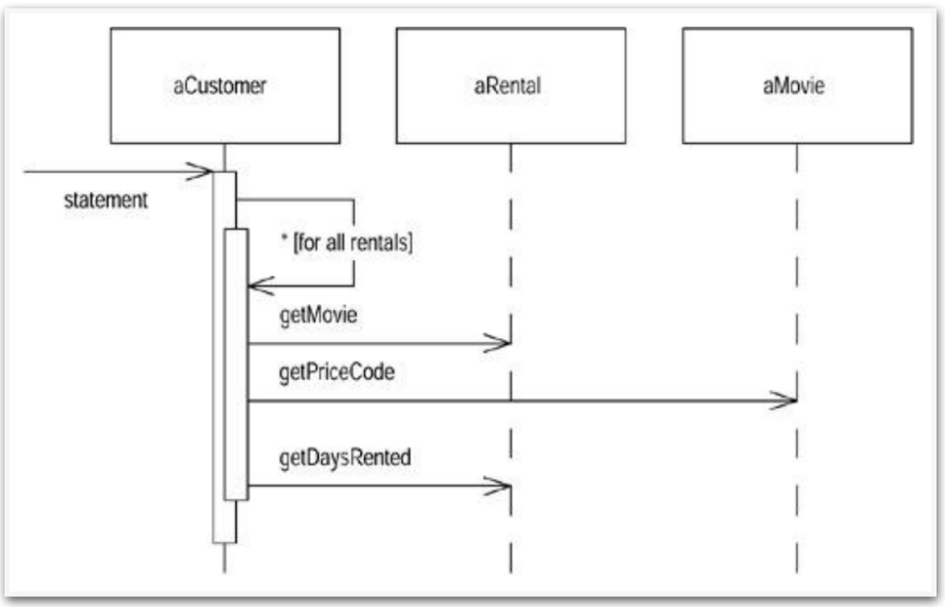


depois

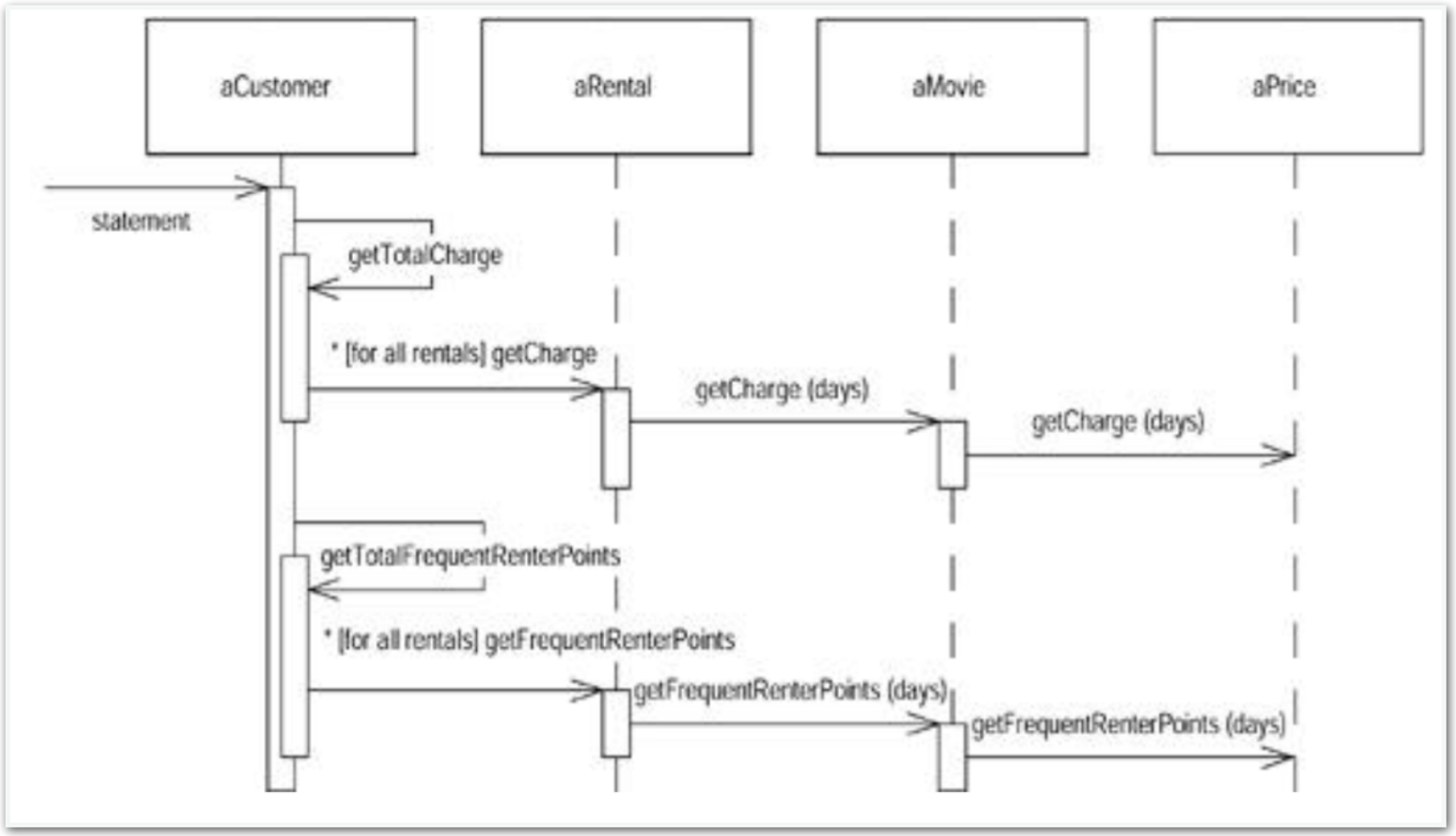


Diagramas de Sequência

antes

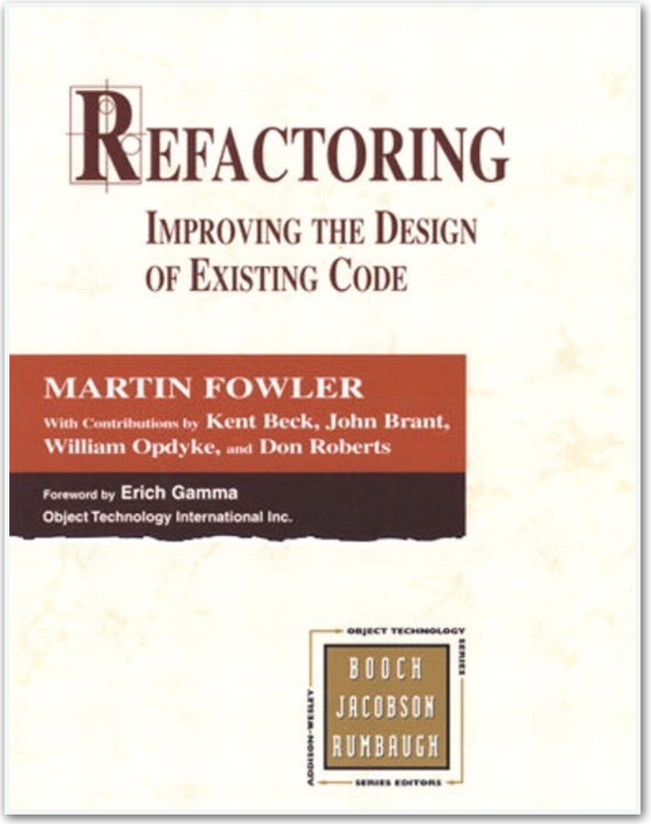


depois



Refatotações Utilizadas

- **Livro**: *Refactoring Improving the Design of Existing Code*



1. Extract Method
2. Move Method
3. Replace Temp with Query
4. Replace Type Code with State/Strategy Pattern
5. Replace Conditional with Polymorphism

Catálogo de Refatorações

1. Add Parameter
2. Change Bidirectional Association to Unidirectional
3. Change Reference to Value
4. Change Unidirectional Association to Bidirectional
5. Change Value to Reference
6. Collapse Hierarchy
7. Consolidate Conditional Expression
8. Consolidate Duplicate Conditional Fragments
9. Decompose Conditional
10. Duplicate Observed Data
11. Encapsulate Collection
12. Encapsulate Downcast
13. Encapsulate Field
14. Extract Class
15. Extract Interface
16. Extract Method
17. Extract Subclass
18. Extract Superclass
19. Extract Variable
20. Form Template Method
21. Hide Delegate
22. Hide Method
23. Inline Class
24. Inline Method
25. Inline Temp
26. Introduce Assertion
27. Introduce Foreign Method
28. Introduce Local Extension
29. Introduce Null Object
30. Introduce Parameter Object
31. Move Field
32. Move Method
33. Parameterize Method
34. Preserve Whole Object
35. Pull Up Constructor Body
36. Pull Up Field
37. Pull Up Method
38. Push Down Field
39. Push Down Method
40. Remove Assignments to Parameters
41. Remove Control Flag
42. Remove Middle Man
43. Remove Parameter
44. Remove Setting Method
45. Rename Method
46. Replace Array with Object
47. Replace Conditional with Polymorphism
48. Replace Constructor with Factory Method
49. Replace Data Value with Object
50. Replace Delegation with Inheritance
51. Replace Error Code with Exception
52. Replace Exception with Test
53. Replace Inheritance with Delegation
54. Replace Magic Number with Symbolic Constant
55. Replace Method with Method Object
56. Replace Nested Conditional with Guard Clauses
57. Replace Parameter with Explicit Methods
58. Replace Parameter with Method
59. Replace Record with Data Class
60. Replace Subclass with Fields
61. Replace Temp with Query
62. Replace Type Code with Class
63. Replace Type Code with State/Strategy
64. Replace Type Code with Subclasses
65. Self Encapsulate Field
66. Separate Query from Modifier
67. Split Temporary Variable
68. Substitute Algorithm

Catálogo de Refatorações

1. Add Parameter
2. Change Bidirectional Association to Unidirectional
3. Change Reference to Value
4. Change Unidirectional Association to Bidirectional
5. Change Value to Reference
6. Collapse Hierarchy
7. Consolidate Conditional Expression
8. Consolidate Duplicate Conditional Fragments
9. Decompose Conditional
10. Duplicate Observed Data
11. Encapsulate Collection
12. Encapsulate Downcast
13. Encapsulate Field
14. Extract Class
15. Extract Interface
16. **Extract Method**
17. Extract Subclass
18. Extract Superclass
19. Extract Variable
20. Form Template Method
21. Hide Delegate
22. Hide Method
23. Inline Class
24. Inline Method
25. Inline Temp
26. Introduce Assertion
27. Introduce Foreign Method
28. Introduce Local Extension
29. Introduce Null Object
30. Introduce Parameter Object
31. Move Field
32. **Move Method**
33. Parameterize Method
34. Preserve Whole Object
35. Pull Up Constructor Body
36. Pull Up Field
37. Pull Up Method
38. Push Down Field
39. Push Down Method
40. Remove Assignments to Parameters
41. Remove Control Flag
42. Remove Middle Man
43. Remove Parameter
44. Remove Setting Method
45. Rename Method
46. Replace Array with Object
47. **Replace Conditional with Polymorphism**
48. Replace Constructor with Factory Method
49. Replace Data Value with Object
50. Replace Delegation with Inheritance
51. Replace Error Code with Exception
52. Replace Exception with Test
53. Replace Inheritance with Delegation
54. Replace Magic Number with Symbolic Constant
55. Replace Method with Method Object
56. Replace Nested Conditional with Guard Clauses
57. Replace Parameter with Explicit Methods
58. Replace Parameter with Method
59. Replace Record with Data Class
60. Replace Subclass with Fields
61. **Replace Temp with Query**
62. Replace Type Code with Class
63. **Replace Type Code with State/Strategy**
64. Replace Type Code with Subclasses
65. Self Encapsulate Field
66. Separate Query from Modifier
67. Split Temporary Variable
68. Substitute Algorithm