

# Refatorações

## Lista 10

### Exercício 1 (2,0)

A partir do código abaixo<sup>1</sup>:

---

```
/* Progama para calcular o valor de 5 Fatorial */

public class Fatorial {

public static void main (String args[]){

double x = 69; // aqui criamos uma variável que irá
    armazenar o numero do fatorial
double f = x; // aqui criamos outra var. Será o
    resultado temporário da multiplicação

while (x > 1){ // Enquanto x for menor que 1 faça o
    que está entre as chaves

    f = f *(x-1); // A variável temporária ira receber
        o resultado da multiplicação dela, pelo valor de
        x menos 1
    x--; // aqui decrementamos o valor de x em um, no
        final do loop
    System.out.println(f); // Esse comando imprime o
        valor de f. O último será o valor final do
        Fatorial.
    }
}
}
```

---

Refatore o programa acertando a indentação, renomeando as variáveis, utilizando um laço for ao invés do while e extraíndo o laço principal do programa para um método estático separado. Escreva testes usando o JUnit para o novo método criado. O método main do seu programa deve apenas chamar o método criado com os devidos parâmetros e exibir na tela o resultado.

---

<sup>1</sup><http://www.devmedia.com.br/calculando-fatorial-em-java/14273>

Trate essa refatoração como uma preparação para a melhoria de funcionalidade: verificação de validade no dado de entrada. aproveite e insira esta melhoria no código.

Nota: seu programa não precisa imprimir os valores intermediários como no código original, apenas o valor final.

## Exercício 2 (1,0)

Dado o código abaixo<sup>2</sup>

---

```
public class AnnualDateRule() {
    /*
     * Construtor padrão.
     */
    protected AnnualDateRule() {
    }
    /** Dia do mês. */
    private int dayOfMonth;
    /**
     * Retorna o dia do mês.
     *
     * @return o dia do mês
     */
    public int getDayOfMonth() {
        return dayOfMonth;
    }
}
```

---

Explique por que o código acima pode ser considerado ruim mesmo contendo diversos comentários.

Dica: use um laço com várias iterações para observar melhor o resultado.

## Exercício 3 (1,0)

Veja os dois códigos abaixo:

---

```
double energiaPotencial(double massa, double altura){
    // 9.81 é a constante gravitacional.
    return massa * 9.81 * altura;
}
```

---

---

```
static final double CONSTANTE_GRAVITACIONAL = 9.81;
double energiaPotencial(double massa, double altura){
    return massa * CONSTANTE_GRAVITACIONAL * altura;
}
```

---

<sup>2</sup><http://pt.slideshare.net/inaelrodrigues1/codigo-limpo-comentarios>

}

---

Explique por que o segundo código é considerado superior.

## Exercício 4 (4,0)

Dado o código abaixo

---

```
package br.usp.ime.refactoring;
import java.text.*;
import java.util.*;
public class CartaoUtil {
    public static final int VISA = 1;
    public static final int MASTERCARD = 2;
    public static final int AMEX = 3;
    public static final int DINERS = 4;
    public static final String CARTAO_OK = "Cartão_
válido";
    public static final String CARTAO_ERRO = "Cartão_
inválido";
    public String validar(int bandeira, String
numero, String validade) {
        boolean validadeOK = false;
        // ----- VALIDADE -----
        Date dataValidade = null;
        try {
            dataValidade = new
                SimpleDateFormat("MM/yyyy").parse(validade);
        } catch (ParseException e) {
            return CARTAO_ERRO;
        }
        Calendar calValidade = new
            GregorianCalendar();
        calValidade.setTime(dataValidade);
        // apenas mês e ano são utilizados na
        validação
        Calendar calTemp = new GregorianCalendar();
        Calendar calHoje = (GregorianCalendar)
            calValidade.clone();
        calHoje.set(Calendar.MONTH,
            calTemp.get(Calendar.MONTH));
        calHoje.set(Calendar.YEAR,
            calTemp.get(Calendar.YEAR));
        validadeOK = calHoje.before(calValidade);
        if (!validadeOK) {
            return CARTAO_ERRO;
        }
    }
}
```

```

}
else {
    // ---- PREFIXO E TAMANHO ----
    String formatado = "";
    // remove caracteres não-numéricos
    for (int i=0; i<numero.length();i++){
        char c=numero.charAt(i);
        if(Character.isDigit(c)){
            formatado +=c;
        }
    }

    boolean formatoOK = false;
    switch (bandeira) {
    case VISA: // tamanhos 13 ou 16, prefixo
        4.
        if (formatado.startsWith("") &&
            (formatado.length() == 13 ||
                                     formatado.length()
                                     ==
                                     16
                                     ))
            {
                formatoOK = true;
            } else {
                formatoOK = false;
            }
        break;
    case MASTERCARD: // tamanho 16, prefixos
        55
        if ((formatado.startsWith("") ||
            formatado.startsWith("") ||
            formatado.startsWith("") ||
            formatado.startsWith("") ||
            formatado.startsWith("") &&
            formatado.length() == 16)) {
            formatoOK = true;
        } else {
            formatoOK = false;
        }
        break;
    case AMEX: // tamanho 15, prefixos 34 e
        37.
        if ((formatado.startsWith("") ||
            formatado.startsWith("") &&
            formatado.length() == 15 )) {

```

```

        formatoOK = true;
    } else {
        formatoOK = false;
    }
    break;
case DINERS: // tamanho 14, prefixos 300
305, 36 e38.
    if ((formatado.startsWith("") ||
        formatoOK = true;
    } else {
        formatoOK = false;
    }
    break;
default:
    formatoOK = false;
    break;
}
if (!formatoOK) {
    return CARTAO_ERRO;
}
else {
    // ----- NÚMERO -----
    // fórmula de LUHN
    (http://www.merriampark.com/anatomycc.htm)
}

int soma = 0;
int digito = 0;
int somafim = 0;
boolean multiplica = false;

for (int i = formatado.length() - 1; i >=
0; i--) {
    digito =
        Integer.parseInt(formatado.substring(i,i+1));
    if (multiplica) {
        somafim = digito * 2;
        if (somafim > 9) {

```

```

        somafim -= 9;
    }
    } else {
        somafim = digito;
    }
    soma += somafim;
    multiplica = !multiplica;
}
int resto = soma % 10;
if (resto == 0) {
    return CARTAO_OK;
} else {
    return CARTAO_ERRO;
}
}
}
}

```

---

Faça a refatoração completa desse pedaço de código. Separe partes do código em funções menores, use polimorfismo para substituir a validação de cartões, etc. Implemente testes usando JUnit para o código criado. Faça pelo menos dois testes de unidade para cada método.

## Exercício 5 (2,0)

Na página

<https://www.cs.virginia.edu/~horton/cs494/s05/slides/lab-exercise-refactoring.htm>

existem diversos exercícios sugeridos de refatoração num código de Monopoly (nosso Banco Imobiliário) disponível nesse mesmo site. Realize dois desses exercícios. Lembre-se de explicitar qual dentre os exercícios descritos na página você resolveu.