Refatorações

Lista 10

Instruções

Instruções de Entrega

Entregar no e-disciplinas DOIS ARQUIVOS.

Nome dos projetos java: ListaX-NUSP1-NUSP2 e L11Exercico05-NUSP1-NUSP2. Sendo X o número da lista para ser realizado em duplas; NUSP1 e NUSP2 são os números usp dos participates.

No primeiro arquivo, você deve entregar os exercícios 1, 2, 3 e 4.

No segundo arquivo, você deve entregar todos os arquivos refatorados do exercício 05.

Entregar os projeto zipados, **apenas** dois projeto por dupla.

Uso de Bibliotecas

Vale a mesma observação da aula anterior, o uso de ferramentas de busca como o Google, a documentação do Java[1], sites sobre programação como o StackOverflow, ou mesmo o Wikipedia será essencial, pois a ideia é vocês pesquisarem como resolver um problema. Por isso, antes de tentarem resolver algum exercício, procurem um pouco e vejam como resolvê-lo de maneira correta.

Pode usar o chatGPT onde for solicitado.

Exercício 1 (2,0)

A partir do código abaixo[2]:

```
/* 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
```

[1] https://docs.oracle.com/javase/17/docs/api/

[2] http://www.devmedia.com.br/calculando-fatorial-em-java/14273

```
            // temporário da multiplicaçã

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 multiplicaçã 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.
                }                                       }
```

Parte A. Refatore o programa acertando a identação, renomeando as variáveis, utilizando um laço for ao invés do `while` e extraindo 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.

Trate essa refatoração como uma preparação para melhoria de funcionalidade: verificação de validade no dado de entrada. Aproveite e insira esta melhoria no código. Seu programa não precisa imprimir os valores intermediários como no código original, apenas o valor final.

Parte B. Usando o ***chatGPT***, solicite a geração de um programa Java semelhante ao de acima, mas com uma modificação: computando a função fatorial de forma decrescente e recursiva, com abundância de comentários e imprimindo valores a cada iteração. Refatore este programa obtido desta maneira, apresentando ambas as versões.

Exercício 2 (1,0)

Dado o código abaixo[3], explique por que este código pode ser considerado ruim mesmo contendo diversos comentários.

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

[3] http://pt.slideshare.net/inaelrodrigues1/codigo-limpo-comentarios

```
        public int getDayofMonth() {
                return dayOfMonth;
        }
}
```

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){
    // g = 9.81: 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;
}
```

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("") ||
  formatado.startsWith("") ||
  formatado.startsWith("") ||
  formatado.startsWith("") ||
  formatado.startsWith("") ||
  formatado.startsWith("") ||
  formatado.startsWith("") &&
  formatado.length() == 14)) {
    formatoOK = true;
  } else {
    formatoOK = false;
  }
  break;
  default:
  formatoOK = false;
  break;
}
if (!formatoOK) {
  return CARTAO_ERRO;
}
else {
  // ----- NÚMERO -----
  // fórmula de LUHN (https://en.wikipedia.org/Luhn_algorithm)

}
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)

Verifique a página `https://www.cs.virginia.edu/~horton/cs494/s05/slides/lab-exercise-refactoring.htm`
e note que existem diversos exercícios sugeridos de refatoração num código de Monopoly (nosso Banco Imobiliário) disponível nesse mesmo site. Antes de iniciar, verifique que o programa está rodando.

Realize as refatorações dos exercícios 2, 3, 4, 5 e 6, entregando apenas o código refatorado. Verifique que o programa deve rodar exatamente como rodava ANTES das refatorações Explioque por que a existências de testes automatizados auxiliariam neste processo. Se quiser, crie os seus testes, ao menos um para cada refatoração.