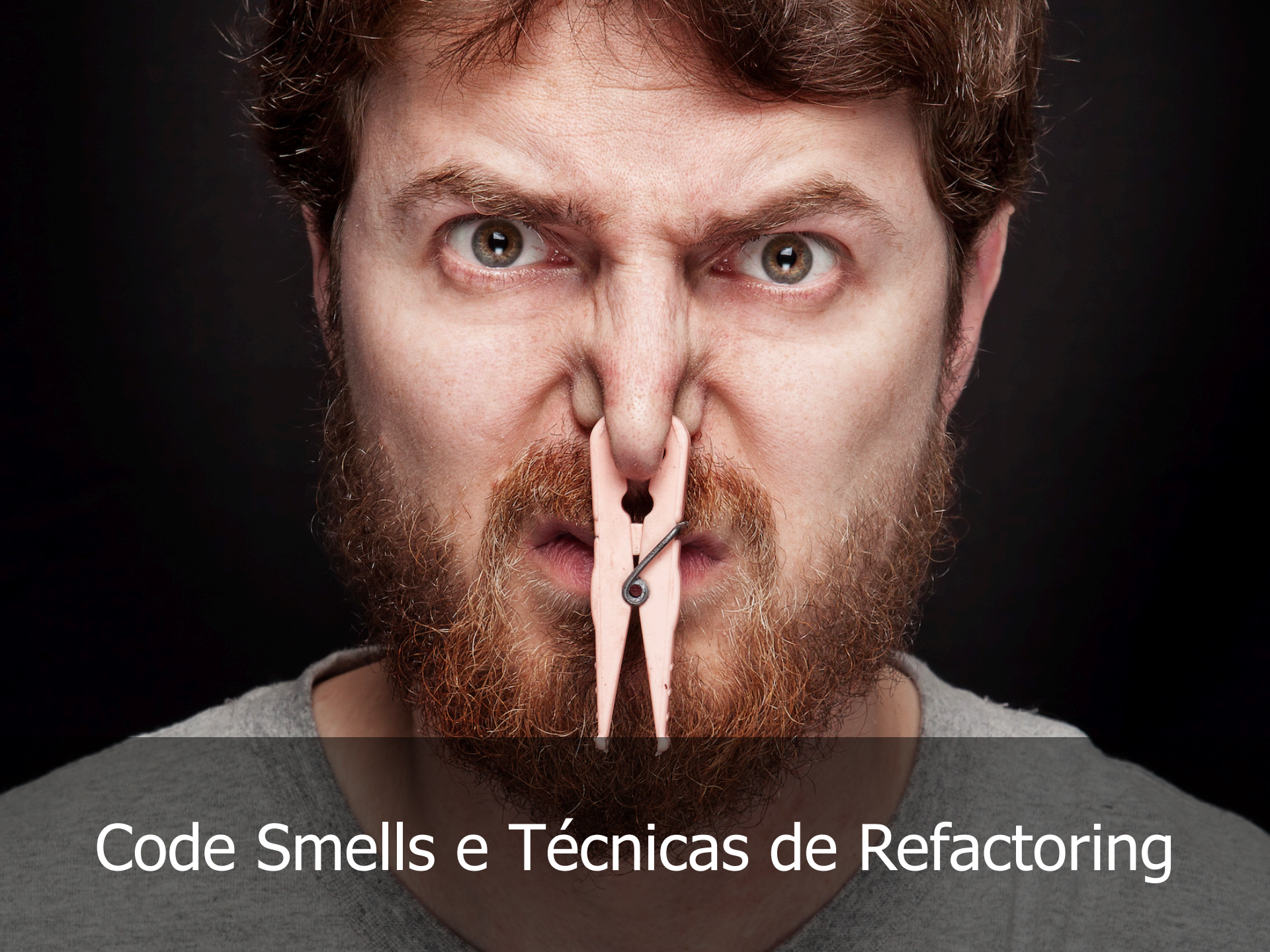
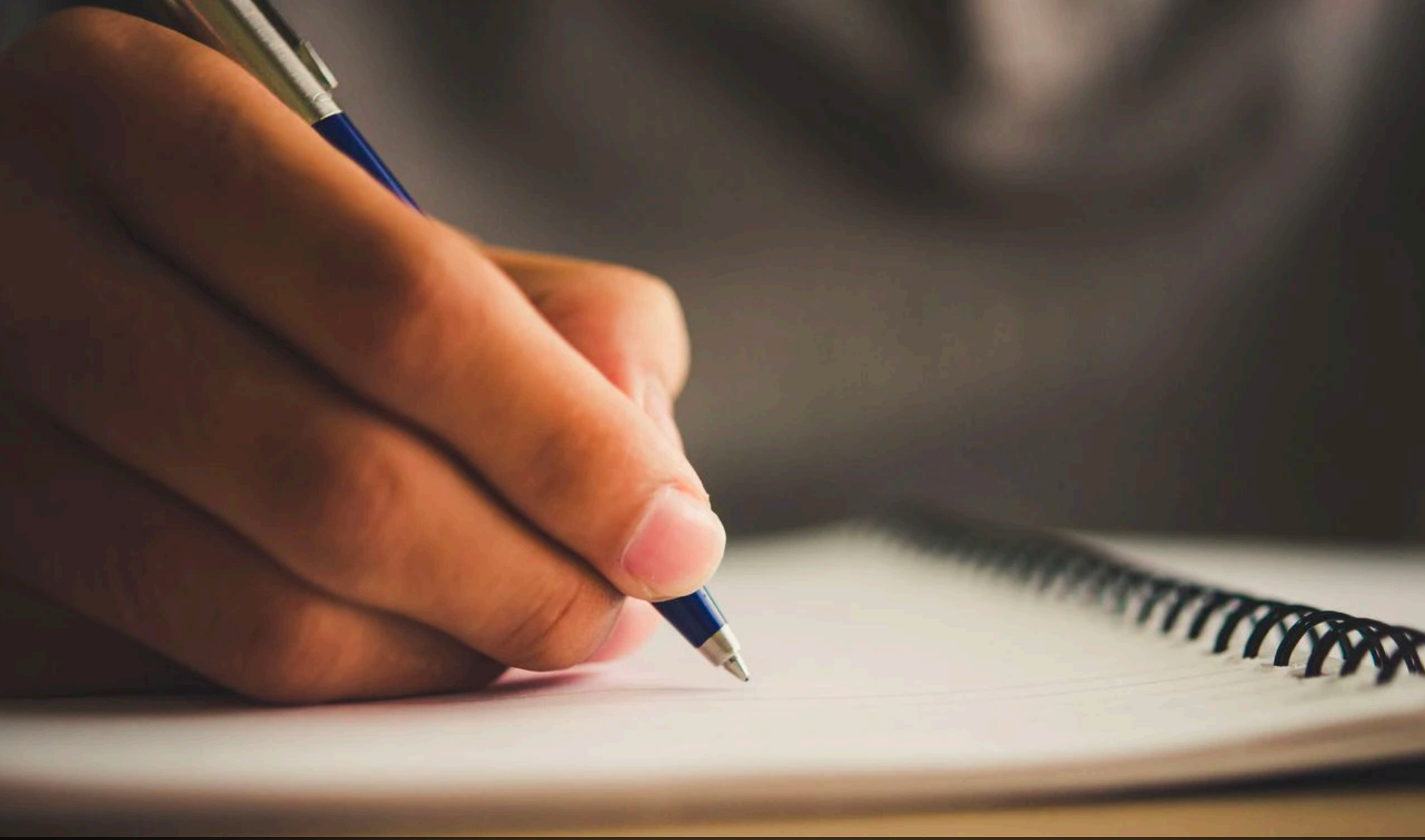


Code Smells e Técnicas de Refactoring



Nomes estranhos

```
public class StringUtil {
    public static String convertAttributeToProperty(String nomeOriginal) {
        String ret = "";
        String aux;
        StringTokenizer token = new StringTokenizer(nomeOriginal, "_");
        ret = token.nextToken();
        while (token.hasMoreTokens()) {
            aux = token.nextToken();
            aux = String.valueOf(aux.charAt(0)).toUpperCase() + aux.substring(1).toLowerCase();
            ret += aux;
        }
        return ret;
    }
```

```
130     */
131    public String getDv43(String numero) {
132
133        int total = 0;
134        int fator = 2;
135
136        int numeros, temp;
137
138        for (int i = numero.length(); i > 0; i--) {
139
140            numeros = Integer.parseInt( numero.substring(i-1,i) );
141
142            temp = numeros * fator;
143            if (temp > 9) temp=temp-9; // Regra do banco NossaCaixa
144
145            total += temp;
146
147            // valores assumidos: 212121...
148            fator = (fator % 2) + 1;
149        }
150
151        int resto = total % 10;
152
153        if (resto > 0)
154            resto = 10 - resto;
155
156        return String.valueOf( resto );
157
158    }
```

Renomear variável

```
1.    function calculateDiscount(amount, p) {
2.        const disc = (amount * p)/100;
3.        return disc;
4.    }
```

```
1.    function calculateDiscount(amount, p) {
2.        const disc = (amount * p)/100;
3.        return disc;
4.    }
```

```
1.    function calculateDiscount(amount, ) {
2.        const disc = (amount * )/100;
3.        return disc;
4.    }
```

```
1.  function calculateDiscount(amount, percentage) {
2.      const disc = (amount * percentage)/100;
3.      return disc;
4.  }
```

```
1.  function calculateDiscount(amount, percentage) {
2.      const disc = (amount * percentage)/100;
3.      return disc;
4.  }
```

```
1.    function calculateDiscount(amount, percentage) {
2.        const  = (amount * percentage)/100;
3.        return ;
4.    }
```

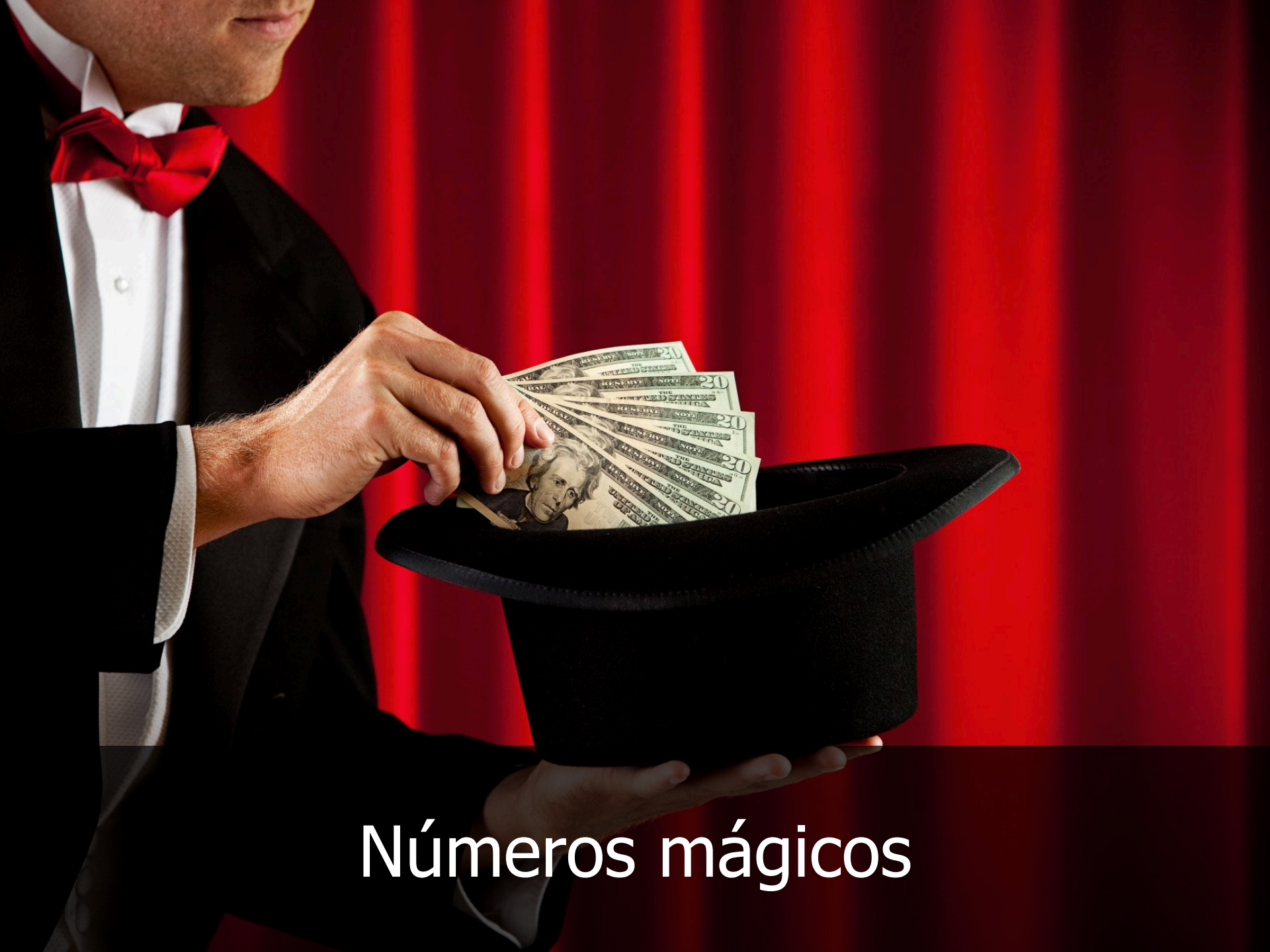
```
1.  function calculateDiscount(amount, percentage) {
2.      const discount = (amount * percentage)/100;
3.      return discount;
4.  }
```

Internalizar Variável Temporária

```
1.  function calculateDiscount(amount, percentage) {
2.      const discount = (amount * percentage)/100;
3.      return discount;
4.  }
```

```
1.  function calculateDiscount(amount, percentage) {
2.      const discount = (amount * percentage)/100;
3.      return discount;
4.  }
```

```
1.    function calculateDiscount(amount, percentage) {
2.        return (amount * percentage)/100;
3.    }
```



Números mágicos

Substituir Números Mágicos por Constantes

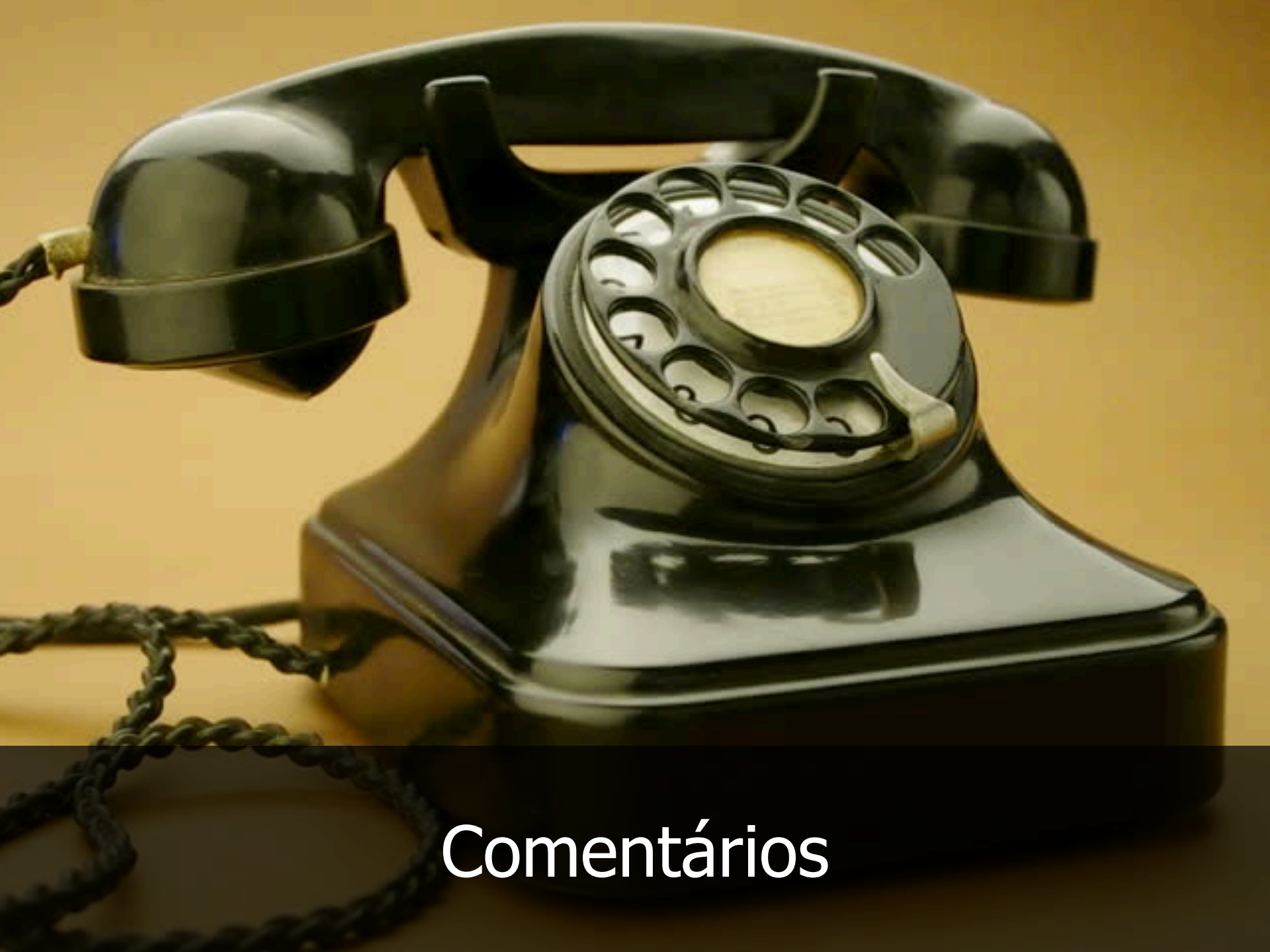
```
1.    function calculatePotencialEnergy(mass, height) {
2.        return mass * 9.81 * height;
3.    }
```

```
1.    function calculatePotencialEnergy(mass, height) {
2.        return mass * 9.81 * height;
3.    }
```

```
1.    function calculatePotencialEnergy(mass, height) {
2.        return mass *  * height;
3.    }
```

```
1.    function calculatePotencialEnergy(mass, height) {
2.        return mass * GRAVITY * height;
3.    }
```

```
1.      const GRAVITY = 9.81;
2.
3.      function calculatePotencialEnergy(mass, height) {
4.          return mass * GRAVITY * height;
5.      }
6.
```



Comentários

```
58          if (nmCache != null && username != null) {
59              // Descriptografa as variáveis
60              nmEntidade = FuncoesUtil.easyDecripto(nmEntidade);
61
62              // Verifica de qual entidade, se deseja recuperar a figura.
63              if (nmEntidade.equalsIgnoreCase("pessoa")) {
64                  // Recupera os dados do request, referentes a entidade em questão.
65                  String idPessoa     = request.getParameter("idPessoa");
66                  String icFuncao     = request.getParameter("icFuncao");
67                  String icTipoPessoa = request.getParameter("icTipoPessoa");
68
69                  // Verifica se existem os parâmetros necessários, no request, referentes a entidade em questão.
70                  if (StringSvc.hasValue(idPessoa) && StringSvc.hasValue(icFuncao)) {
71                      // Descriptografa as variáveis
72                      idPessoa = FuncoesUtil.easyDecripto(idPessoa);
73                      icFuncao = FuncoesUtil.easyDecripto(icFuncao);
74
75                      // Verifica se foi passado o tipo da pessoa, caso tenha, o valor do idPessoa corresponde a u
76                      // professor, aluno, etc. e portando deverá primeiramente obter o código da pessoa correspon
77                      if (StringSvc.hasValue(icTipoPessoa)) {
78                          // Descriptografa as variáveis
79                          icTipoPessoa = FuncoesUtil.easyDecripto(icTipoPessoa);
80
81                          // Recupera o código da pessoa, correspondente a entidade passada (aluno, professor, etc
82                          idPessoa = String.valueOf(getCodigoPessoa(nmCache, username, idPessoa, icTipoPessoa));
83                      }
84
85                      // Verifica se existe o código da pessoa.
86                      if (StringSvc.hasValue(idPessoa)) {
87                          to = new ASPTO();
88                          // Cria um TO, contendo os dados necessários para a recuperação da entidade Pessoa
89                          to.addColumn("idPessoa", "Integer");
90                          // O campo icFuncao indica Tipo da figura a ser recuperada (1 - Assinatura, 2 - Foto)
91                          to.addColumn("icFuncao", "Integer");
92
93                          to.setValue("idPessoa", new Integer(idPessoa));
94                          to.setValue("icFuncao", new Integer(icFuncao));
95                      }
96
97                      // Verifica se existe campos no TO.
98                      if (to != null) {
99                          retorno = getData(nmCache, username, nmEntidade, to);
```

```
40          ActionForward forward           = null;
41          ASPMessageTO messages           = new ASPMessageTO();
42
43          // Verifica se o usuário tem permissão de entrar na página
44          forward = hasPermissao(mapping, request, "quadroHorario.vm", "quadroHorario.titulo");
45          if (forward == null) {
46              // Chama método que recupera o quadro de horário tratando retorno de erros
47              messages = getQuadroHorarioAluno(request);
48
49              // Seta forward
50              forward = mapping.findForward("quadroHorario");
51              saveAllMessages(request, messages);
52          }
53          return forward;
54
55
```

Introduzir Variável Explicativa

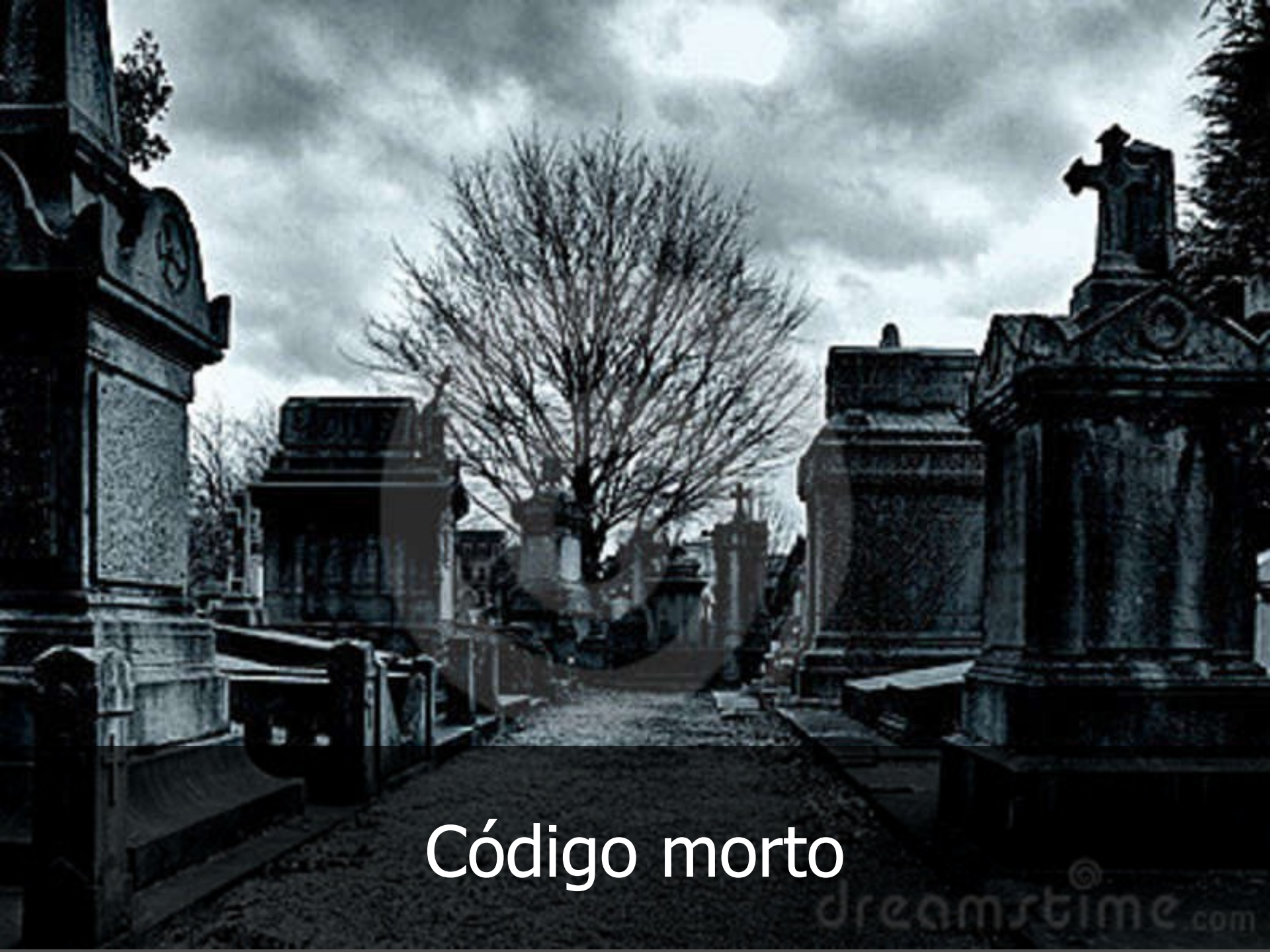
```
1.  function calculateRide(hour, distance) {
2.      if (hour > 22 || hour < 6) {
3.          return distance * 3.90;
4.      } else {
5.          return distance * 2.10;
6.      }
7.  }
```

```
1.  function calculateRide(hour, distance) {
2.      if (hour > 22 || hour < 6) {
3.          return distance * 3.90;
4.      } else {
5.          return distance * 2.10;
6.      }
7.  }
```

```
1.  function calculateRide(hour, distance) {
2.      if () {
3.          return distance * 3.90;
4.      } else {
5.          return distance * 2.10;
6.      }
7.  }
```

```
1.  function calculateRide(hour, distance) {
2.      if (isOvernight) {
3.          return distance * 3.90;
4.      } else {
5.          return distance * 2.10;
6.      }
7.  }
```

```
1.  function calculateRide(hour, distance) {
2.      const isOvernight = hour > 22 || hour < 6;
3.      if (isOvernight) {
4.          return distance * 3.90;
5.      } else {
6.          return distance * 2.10;
7.      }
8.  }
```


Código morto

Apagar o código

```
3212
3213                    if(to.getStringValue("nrCpf") == null)
3214                        txtLinha.append(";");
3215                    else
3216                        txtLinha.append(to.getStringValue("nrCpf").trim() + ";");
3217
3218                    txtLinha.append(to.getValue("nmAluno") + ";");
3219
3220                    if(to.getStringValue("nrIdentidade") == null)
3221                        txtLinha.append(";");
3222                    else
3223                        txtLinha.append(to.getStringValue("nrIdentidade").trim() + ";");
3224
3225                    txtLinha.append(to.getValue("idDeficienciaFisica") + ";");
3226                    txtLinha.append(to.getValue("idDeficienciaVisual") + ";");
3227                    txtLinha.append(to.getValue("idDeficienciaAuditiva") + ";");
3228
3229 /                  txtLinha.append(request.getParameter("cdHabilitacaoEnade") + ";");
3230 /                  if(request.getParameter("icAlunos") == "1")
3231 /                      txtLinha.append("0" + ";");
3232 /                  else
3233 /                      txtLinha.append("1" + ";");
3234 /                  if(to.getIntegerValue("idSexo") == 0)
3235 /                      txtLinha.append(";");
3236 /                  else
3237 /                      txtLinha.append(to.getValue("idSexo") + ";");
3238
3239                    if(to.getIntegerValue("cdCep") == 0)
3240                        txtLinha.append(";");
3241                    else
3242                        txtLinha.append(to.getValue("cdCep") + ";");
3243
3244                    if(to.getStringValue("dsLogradouro") == null)
3245                        txtLinha.append(";");
3246                    else {
3247                        if(to.getStringValue("dsLogradouro").trim().length() > 60)
3248                            txtLinha.append(to.getStringValue("dsLogradouro").trim().substring(0,60) + ";");
3249                        else
3250                            txtLinha.append(to.getValue("dsLogradouro") + ";");
3251                    }
3252
3253                    if(request.getParameter("numero") == null)
```

```
1088        // Verifica dados obrigatórios das pastas (curso e critério).
1089        if (StringSvc.hasValue(janelaOrigem)) {
1090            if (janelaOrigem.equalsIgnoreCase("registro")) {
1091                messages.addMessages(verificaDadosObrigatoriosPastaRegistro(actionForm, request, metodo));
1092            } else if (janelaOrigem.equalsIgnoreCase("assunto")) {
1093                //messages.addMessages(verificaDadosObrigatoriosPastaAssunto(actionForm, request, metodo));
1094            } else if (janelaOrigem.equalsIgnoreCase("anexo")) {
1095                //messages.addMessages(verificaDadosObrigatoriosPastaAnexo(actionForm, request, metodo));
1096            }
1097        }
1098        return messages;
1099    }
1100
1101    /**
```

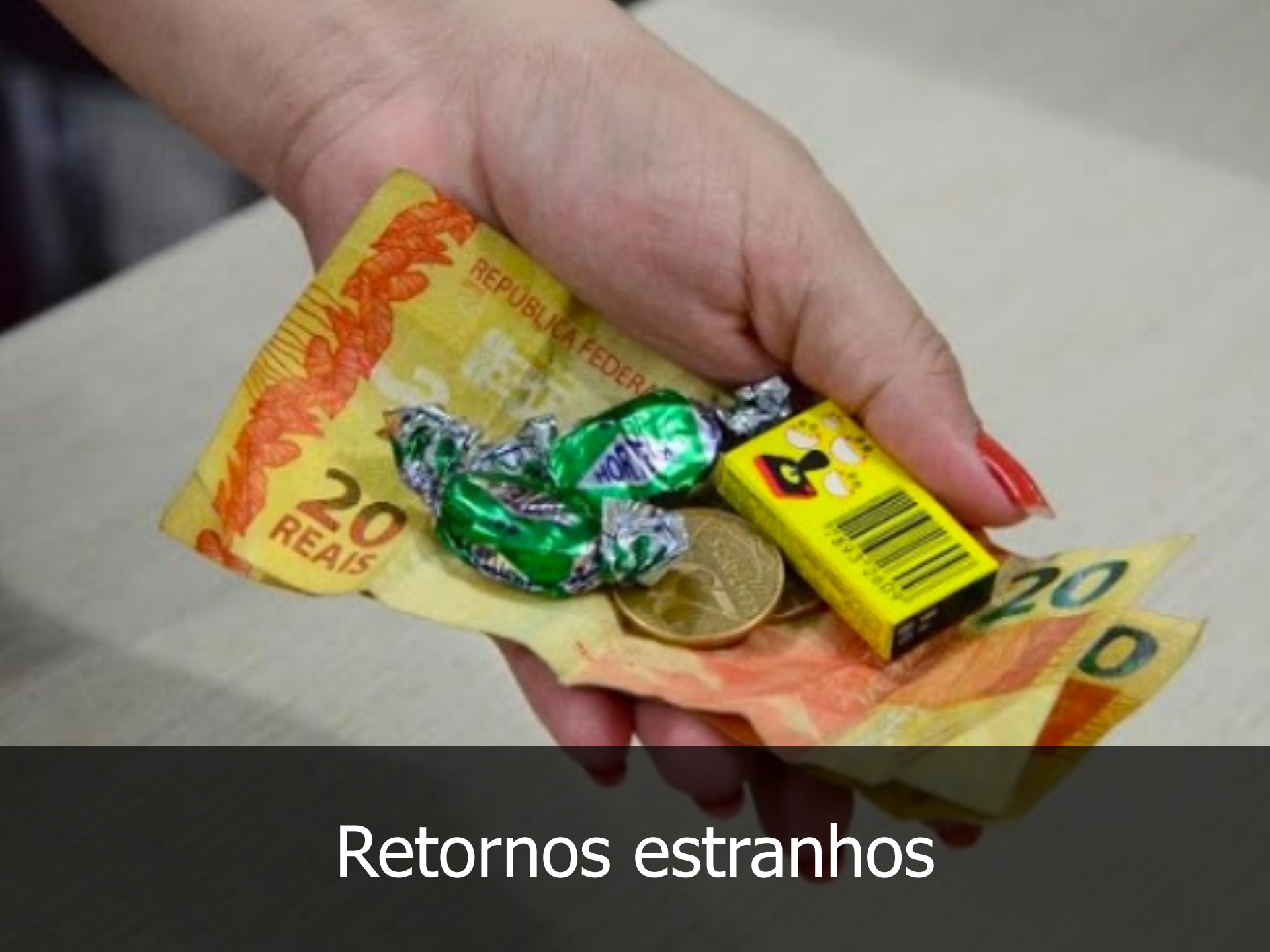
```
141             outStream.write(fileBlob);
142         } else {
143             response.setContentType("text/html");
144             outStream.println("<HTML>");
145             outStream.println("<HEAD>");
146             outStream.println("    <META NAME=SERVERLANGUAGE CONTENT=JavaScript HTTP-EQUIV=PowerSiteData>");
147             outStream.println("</HEAD>");
148             outStream.println("<HEAD>");
149             outStream.println("    <TITLE>Download de Arquivos</TITLE>");
150             outStream.println("    <LINK HREF=\"./portal/css/hsEstilos.css\" REL=stylesheet>");
151             outStream.println("    <SCRIPT LANGUAGE=javascript SRC=\"./portal/js/funcoes.js\" SCRIPT></SCRIPT>");
152             outStream.println("</HEAD>");
153             outStream.println("<BODY>");
154             outStream.println("<SCRIPT LANGUAGE=\"JavaScript1.2\" TYPE=\"text/javascript\" SRC=\"./portal/js/cframe.js\"
155             //      outStream.println("<a name=\"top\"></a>");
156             //      outStream.println("<table width=\"595\" border=\"0\" cellspacing=\"0\" cellpadding=\"0\" align=\"cen
157             //      outStream.println("    <tr> ");
158             //      outStream.println("<TD width=27><IMG height=39 src=\"./portal/images/mn_inicio.gif\" width=27></TD>"
159             //      outStream.println("<TD class=mnAzul background=\"./portal/images/mn_fundo.gif\" width=403>");
160             //      outStream.println((titlePage != null ? titlePage : "MANUTENÇÂO :: <STRONG>Geração de arquivo</STRONG
161             //      outStream.println("            <TD align=\"right\" background=\"./portal/images/mn_fundo.gif\" width
162             //      outStream.println("</tr>");
163             //      outStream.println("</table>");
164             //      outStream.println("    <TABLE border=0 align=center cellSpacing=0 cellPadding=0 width=595>");
165             //      outStream.println("        <TBODY>");
166             //      outStream.println("        <TR>");
167             //      outStream.println("            <TD><IMG src=\"./portal/images/j_fundo2.gif\"> </TD>");
168             //      outStream.println("        </TR>");
169             //      outStream.println("        <TR>");
170             //      outStream.println("            <TD class=txtNormal width=462><br>" + messageResult + "</TD>");
171             //      outStream.println("        </TR>");
172             //      outStream.println("        <TR>");
173             //      outStream.println("            <TD><IMG src=\"./portal/images/j_fundo2.gif\"> </TD>");
174             //      outStream.println("        </TR>");
175             //      outStream.println("        </TBODY>");
176             //      outStream.println("    </TABLE>");
177             outStream.println("<script>");
178             outStream.println("alert('" + upload.strTran(messageResult, "\n", "") + "')");
179             outStream.println("</script>");
180             outStream.println("</body>");
181             outStream.println("</HTML>");
182         }
183     }
```

```
220         }
221
222         if ((new Integer(icDigitacao)).intValue() == 1) {
223             // Fazer dentro de if para não criar uma referencia ao invés de uma nova cópia.
224             if (true) {
225                 ASPDynaTO[] dynaVectorTO      = ((ASPDynaTO[])actionForm.get("notas"));
226                 ASPDynaTO[] dynaVectorTOClone = dynaVectorTO.clone();
227                 actionForm.set("notasPrimeira", dynaVectorTOClone);
228             }
229
230             if (true) {
231                 ASPDynaTO[] dynaVectorTO      = ((ASPDynaTO[])actionForm.get("notasOriginal"));
232                 ASPDynaTO[] dynaVectorTOClone = dynaVectorTO.clone();
233                 actionForm.set("notas", dynaVectorTOClone);
234             }
235         } else {
236             ASPDynaTO[] dynaVectorTO      = ((ASPDynaTO[])actionForm.get("notasPrimeira"));
237             ASPDynaTO[] dynaVectorTOClone = dynaVectorTO.clone();
238             actionForm.set("notas", dynaVectorTOClone);
```

Linhas em branco

Apagar as linhas em branco

```
130     */
131    public String getDv43(String numero) {
132
133        int total = 0;
134        int fator = 2;
135
136        int numeros, temp;
137
138        for (int i = numero.length(); i > 0; i--) {
139
140            numeros = Integer.parseInt( numero.substring(i-1,i) );
141
142            temp = numeros * fator;
143            if (temp > 9) temp=temp-9; // Regra do banco NossaCaixa
144
145            total += temp;
146
147            // valores assumidos: 212121...
148            fator = (fator % 2) + 1;
149        }
150
151        int resto = total % 10;
152
153        if (resto > 0)
154            resto = 10 - resto;
155
156        return String.valueOf( resto );
157
158    }
```



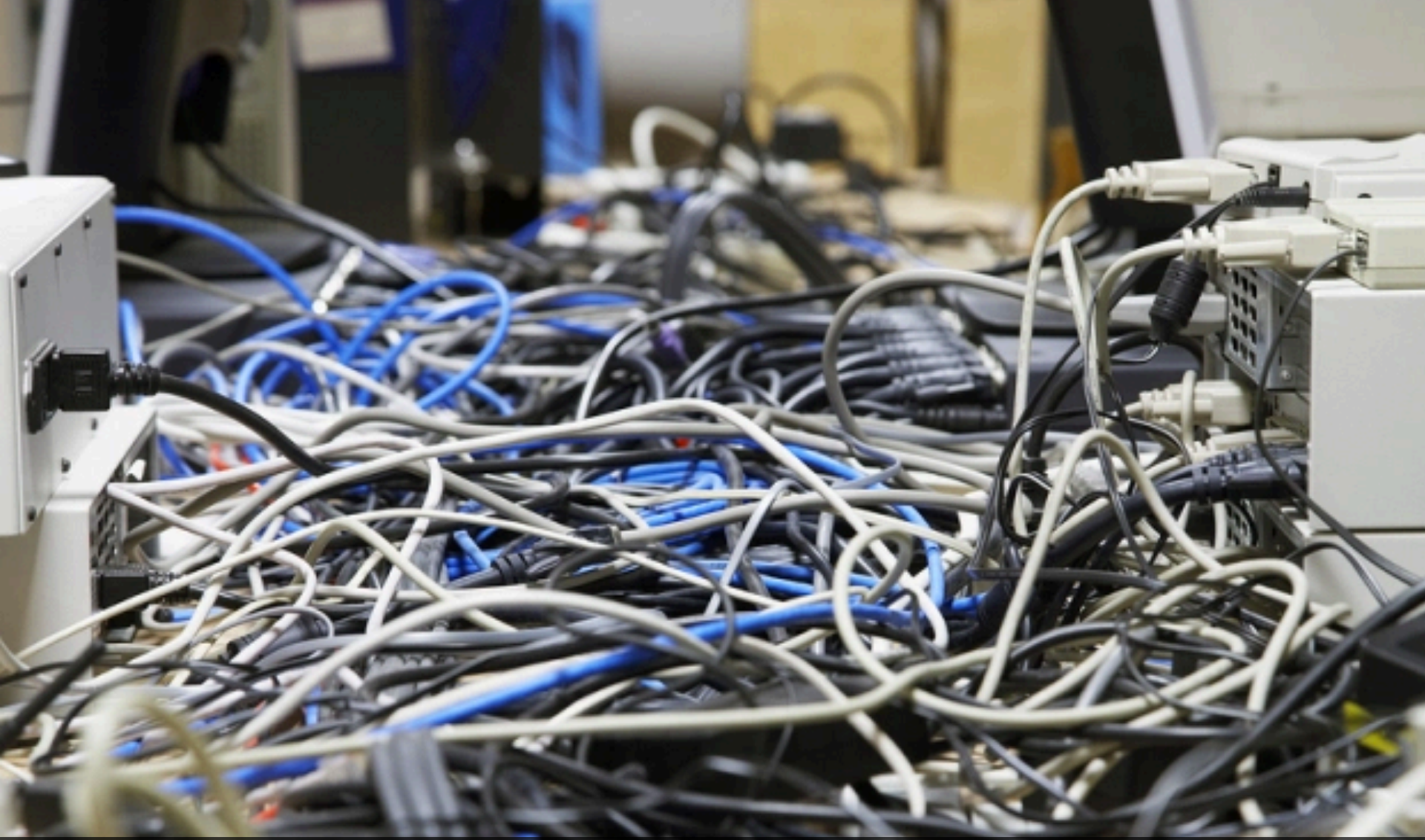
Retornos estranhos

```
1379            int b;
1380            for( i = 0; i < len; i++ ) {
1381                b = read();
1382
1383                //if( b < 0 && i == 0 )
1384                //    return -1;
1385
1386                if( b >= 0 )
1387                    dest[off + i] = (byte)b;
1388                else if( i == 0 )
1389                    return -1;
1390                else
1391                    break; // Out of 'for' loop
1392            }   // end for: each byte read
1393            return i;
1394        }   // end read
1395
```

Substituir código de erro por exceção

```
1.      class WithdrawMoney {
2.
3.        async execute (account, amount) {
4.          const result = await debitAccount.execute(account, amount);
5.          if (result === 1) {
6.            try {
7.              await atm.dispenseMoney(amount);
8.            } catch (e) {
9.              await creditAccount.execute(account, amount);
10.           }
11.         }
12.         if (result === -1) {
13.           await atm.showMessage("Insuficient balance");
14.         }
15.         if (result === -2) {
16.           await atm.showMessage("Card has expired");
17.         }
18.       }
19.     }
```

```
1.      class WithdrawMoney {
2.
3.        async execute (account, amount) {
4.          try {
5.            await debitAccount.execute(account, amount);
6.            await atm.dispenseMoney(amount);
7.          } catch (e) {
8.            await creditAccount.execute(account, amount);
9.            await atm.showMessage(e.message);
10.         }
11.       }
12.     }
```



Condições confusas

```
2875            }
2876        } catch(Exception e){
2877            e.printStackTrace();
2878        }
2879
2880        // Turmas
2881        xmlLista = new XMLSvc();
2882        try {
2883            comp = (AcademicoDelegate) Funcoes.createDelegate(AcademicoDelegateHome.JNDI_NAME, AcademicoDelegateHome
2884            ResultTO result = comp.execute(nmCache, cdUsuario, "exportacaoCursoTurma", new Object[]{whereClause.toSt
2885
2886            if (result.ok) {
2887                list = (ASPVector)result.value;
2888                if (list != null && list.size() > 0) {
2889                    for (int i = 0; i < list.size(); i++) {
2890                        XMLSvc xmlLinha = new XMLSvc();
2891                        ASPTO to = (ASPTO)list.get(i);
2892                        String codigoTurma = to.getValue("cdCursoInstituicao") + "-" + to.getValue("cdTurma").toStri
2893                        xmlLinha.addTag("CODIGO", codigoTurma);
2894                        if (StringSvc.hasValue(to.getStringValue("dsTurma"))){
2895                            xmlLinha.addTag("NOME", to.getStringValue("dsTurma"));
2896                        }
2897                        // Os campos DIAS e AULAS do layout da TURMA estão sendo desprezados  na integração
2898                        //xmlLinha.addTag("DIAS", "");
2899                        //xmlLinha.addTag("AULAS", "");
2900                        xmlLinha.addTag("ABREVIATURA", codigoTurma);
2901                        xmlLinha.addTag("TURNO", to.getStringValue("dsTurno"));
2902                        xmlLista.addTag("REGISTRO", xmlLinha.getFragment());
2903                    }
2904                    xml.addTag("TURMAS", xmlLista.getFragment());
2905                }
2906            }
2907        } catch(Exception e){
2908            e.printStackTrace();
2909        }
2910
2911        // Professores
2912        xmlLista = new XMLSvc();
2913        list = getListTO(ProfessorLocalHome.JNDI_NAME, nmCache, cdUsuario, whereClause.toString(), "professor.cd_pro
2914        if (list != null && list.size() > 0) {
2915            for (int i = 0; i < list.size(); i++) {
2916                XMLSvc xmlLinha = new XMLSvc();
2917                ASPTO to = (ASPTO)list.get(i);
```

```
                }
                //Pesquisa os Professores realcionados as perguntas
                searchPesquisaPerguntaProfessor(request,pesquisa.getIntegerValue("cdPesquisa"),actionForm);
            }
        }


        if (pesquisaRestrita.size() > 0) {
            ASPVector pesquisas = pesquisaRestrita;

            if(quadroHorario.size() > 0){
                for (int i = 0; i < quadroHorario.size(); i++) {
                    QuadroHorarioAlunoTO quadroHorarioAluno = (QuadroHorarioAlunoTO) quadroHorario.get(i);

                    for (int j = 0; j < pesquisas.size(); j++) {
                        PesquisaSocioEconomicaTO pesquisa = (PesquisaSocioEconomicaTO) pesquisas.get(j);
                        ASPVector questionario = pesquisa.getASPVectorValue("questionario");
                        ASPVector questionarioCorrigido = new ASPVector();

                        boolean possuiProfessor = false;
                        for (int k = 0; k < questionario.size(); k++) {
                            PesquisaPerguntaQuestionarioTO perguntaQuestionario = (PesquisaPerguntaQuestionarioTO
                            if(perguntaQuestionario.getIntegerValue("cdProfessor") != null){
                                possuiProfessor = true;
                                break;
                            }
                        }
                        if(possuiProfessor){
                            for (int k = 0; k < questionario.size(); k++) {
                                PesquisaPerguntaQuestionarioTO perguntaQuestionario = (PesquisaPerguntaQuestionar
                                if(perguntaQuestionario.getIntegerValue("cdProfessor").intValue() == Integer.pars
                                    questionarioCorrigido.add(perguntaQuestionario);
                                }
                            }
                            if(questionarioCorrigido.size() > 0){
                                pesquisa.setValue("questionario", questionarioCorrigido);
                            }
                        }
                    }
                }
                request.getSession().setAttribute("ASPPesquisas", pesquisas);
            }else{
```

```
159                     result = comp.execute(nmCache, cdUsuario, "pesquisaNotasParciais", new Object[] { whereClause, ""  }, NotaLocal
160                     MessageUtil.actionMessages(messages, result.bufferMessage);
161                     if (result.ok) {
162                         list = (ASPVector)result.value;
163                         listCarga = (ASPVector)result.value;
164                         if(list.size() > 0){
165                             int cdDisciplinaLista = 0;
166                             to = list.getTO(0);
167                             cdDisciplinaLista = to.getIntegerValue("cdDisciplina").intValue();
168                             dynaTO = ASPDynaTO.getInstance(NotasParciaisTO.getOriginalMetaData());
169 
170                             if(to.getIntegerValue("qtCargaHoraria") != null){
171                                 cargaHoraria = cargaHoraria + to.getIntegerValue("qtCargaHoraria").intValue();
172                                 pctFreque = pctFreque + to.getDoubleValue("pctFrequencia");
173                             }else{
174                                 cargaHoraria = 0;
175                                 pctFreque = new Double(0);
176                             }
177                             //Soma de carga horária de todos os periodos habilitados no acesso modulo agregado
178                             for (int i = 1; i < list.size(); i++) {
179                                 to = list.getTO(i);
180                                 if(to.getIntegerValue("qtCargaHoraria") != null){
181                                     if(cdDisciplinaLista == to.getIntegerValue("cdDisciplina").intValue() ){
182                                         if(to.getIntegerValue("cdPeriodo").intValue() <= cdPeriodo){
183                                             cargaHoraria = cargaHoraria + to.getIntegerValue("qtCargaHoraria").intValue();
184                                         }
185                                     }else{
186                                         toCarga = list.getTO(i - 1);
187                                         toCarga.setValue("qtCargaHoraria", cargaHoraria);
188                                         for (int j = 0; j < listCarga.size(); j++) {
189                                             toCarga = list.getTO(j);
190                                             if(cdDisciplinaLista == toCarga.getIntegerValue("cdDisciplina").intValue() ){
191                                                 toCarga.setValue("qtCargaHoraria", cargaHoraria);
192                                             }
193                                         }
194                                         cargaHoraria = 0;
195                                         cargaHoraria = cargaHoraria + to.getIntegerValue("qtCargaHoraria").intValue();
196                                         cdDisciplinaLista = to.getIntegerValue("cdDisciplina").intValue();
197                                     }
198                                 }else{
199                                     to.setValue("qtCargaHoraria", new Integer(0));
200                                 }
201                             }
202                             for (int j = 0; j < listCarga.size(); j++) {
203                                 toCarga = list.getTO(j);
204                                 if(cdDisciplinaLista == toCarga.getIntegerValue("cdDisciplina").intValue() ){
205                                     toCarga.setValue("qtCargaHoraria", cargaHoraria);
206                                 }
207                             }
208 
209                             //Soma do percentual de frequencia de até periodo
210                             cdDisciplinaLista = new Integer(0);
211                             for (int j = 0; j < list.size(); j++) {
```

```
1266     public int read() throws java.io.IOException {
1267         // Do we need to get data?
1268         if( position < 0 ) {
1269             if( encode ) {
1270                 byte[] b3 = new byte[3];
1271                 int numBinaryBytes = 0;
1272                 for( int i = 0; i < 3; i++ ) {
1273                     try {
1274                         int b = in.read();
1275
1276                         // If end of stream, b is -1.
1277                         if( b >= 0 ) {
1278                             b3[i] = (byte)b;
1279                             numBinaryBytes++;
1280                         }   // end if: not end of stream
1281
1282                     }   // end try: read
1283                     catch( java.io.IOException e ) {
1284                         // Only a problem if we got no data at all.
1285                         if( i == 0 )
1286                             throw e;
1287
1288                     }   // end catch
1289                 }   // end for: each needed input byte
1290
1291                 if( numBinaryBytes > 0 ) {
1292                     encode3to4( b3, 0, numBinaryBytes, buffer, 0, options );
1293                     position = 0;
1294                     numSigBytes = 4;
1295                 }   // end if: got data
1296                 else {
1297                     return -1;
1298                 }   // end else
1299             }   // end if: encoding
1300
1301             // Else decoding
1302             else {
1303                 byte[] b4 = new byte[4];
1304                 int i = 0;
1305                 for( i = 0; i < 4; i++ ) {
1306                     // Read four "meaningful" bytes:
1307                     int b = 0;
```

Remover condição aninhada por cláusulas guarda

```
async function sendEmailCampaign(campaign) {
  if (!campaign.sent) {
    const recipients = await repository.getRecipients(campaign.id);
    if (hasEmailQuota(recipients.length)) {
      for (const recipient of recipients) {
        if (!isBouncedRecipient(recipient)) {
          if (isAllowedRecipient(recipient)) {
            await send(campaign, recipient);
          }
        }
      }
    }
  }
}
```

```
1.  async function sendEmailCampaign(campaign) {
2.    if (!campaign.sent) {
3.      const recipients = await repository.getRecipients(campaign.id);
4.      if (hasEmailQuota(recipients.length)) {
5.        for (const recipient of recipients) {
6.          if (!isBouncedRecipient(recipient)) {
7.            if (isAllowedRecipient(recipient)) {
8.              await send(campaign, recipient);
9.            }
10.         }
11.       }
12.     }
13.   }
14. }
```

```
1.  async function sendEmailCampaign(campaign) {
2.    if (campaign.sent) return;
3.    const recipients = await repository.getRecipients(campaign.id);
4.    if (hasEmailQuota(recipients.length)) {
5.      for (const recipient of recipients) {
6.        if (!isBouncedRecipient(recipient)) {
7.          if (isAllowedRecipient(recipient)) {
8.            await send(campaign, recipient);
9.          }
10.       }
11.     }
12.   }
13. }
```

```
1.  async function sendEmailCampaign(campaign) {
2.    if (campaign.sent) return;
3.    const recipients = await repository.getRecipients(campaign.id);
4.    if (hasEmailQuota(recipients.length)) {
5.      for (const recipient of recipients) {
6.        if (!isBouncedRecipient(recipient)) {
7.          if (isAllowedRecipient(recipient)) {
8.            await send(campaign, recipient);
9.          }
10.       }
11.     }
12.   }
13. }
```

```
1.  async function sendEmailCampaign(campaign) {
2.    if (campaign.sent) return;
3.    const recipients = await repository.getRecipients(campaign.id);
4.    if (!hasEmailQuota(recipients.length)) return;
5.    for (const recipient of recipients) {
6.      if (!isBouncedRecipient(recipient)) {
7.        if (isAllowedRecipient(recipient)) {
8.          await send(campaign, recipient);
9.        }
10.     }
11.   }
12. }
```

```
1.  async function sendEmailCampaign(campaign) {
2.    if (campaign.sent) return;
3.    const recipients = await repository.getRecipients(campaign.id);
4.    if (!hasEmailQuota(recipients.length)) return;
5.    for (const recipient of recipients) {
6.      if (!isBouncedRecipient(recipient)) {
7.        if (isAllowedRecipient(recipient)) {
8.          await send(campaign, recipient);
9.        }
10.     }
11.   }
12. }
```

```
1.  async function sendEmailCampaign(campaign) {
2.    if (campaign.sent) return;
3.    const recipients = await repository.getRecipients(campaign.id);
4.    if (!hasEmailQuota(recipients.length)) return;
5.    for (const recipient of recipients) {
6.      if (isBouncedRecipient(recipient)) continue;
7.      if (isAllowedRecipient(recipient)) {
8.        await send(campaign, recipient);
9.      }
10.   }
11. }
```

```
1.  async function sendEmailCampaign(campaign) {
2.    if (campaign.sent) return;
3.    const recipients = await repository.getRecipients(campaign.id);
4.    if (!hasEmailQuota(recipients.length)) return;
5.    for (const recipient of recipients) {
6.      if (isBouncedRecipient(recipient)) continue;
7.      if (isAllowedRecipient(recipient)) {
8.        await send(campaign, recipient);
9.      }
10.   }
11. }
```

```
async function sendEmailCampaign(campaign) {
  if (campaign.sent) return;
  const recipients = await repository.getRecipients(campaign.id);
  if (!hasEmailQuota(recipients.length)) return;
  for (const recipient of recipients) {
    if (isBouncedRecipient(recipient)) continue;
    if (!isAllowedRecipient(recipient)) continue;
    await send(campaign, recipient);
  }
}
```

Consolidar fragmentos condicionais duplicados

```
async function processPostback(gateway, info) {
  if (gateway.type === "stripe") {
    if (info.status === "succeeded") {
      const trackNumber = info.data.object.id;
      const order = await repository.getOrderByTrackNumber(trackNumber);
      await confirmOrder.execute(order.id);
    }
  }
  if (gateway.type === "paypal") {
    if (info.status === "APPROVED") {
      const trackNumber = info.resource.id;
      const order = await repository.getOrderByTrackNumber(trackNumber);
      await confirmOrder.execute(order.id);
    }
  }
}
```

```
async function processPostback(gateway, info) {
  let order;
  if (gateway.type === "stripe") {
    if (info.status === "succeeded") {
      const trackNumber = info.data.object.id;
      order = await repository.getOrderByTrackNumber(trackNumber);
    }
  }
  if (gateway.type === "paypal") {
    if (info.status === "APPROVED") {
      const trackNumber = info.resource.id;
      order = await repository.getOrderByTrackNumber(trackNumber);
    }
  }
  await confirmOrder.execute(order.id);
}
```

Consolidar expressão condicional

```
async function processPostback(gateway, info) {
  if (gateway.type === "paypal") {
    if (info.status === "APPROVED") {
      await processPaypal(info);
    }
  }
}
```

```
async function processPostback(gateway, info) {
  if (gateway.type === "paypal" && info.status === "APPROVED") {
    await processPaypal(info);
  }
}
```

Introduzir comando ternário

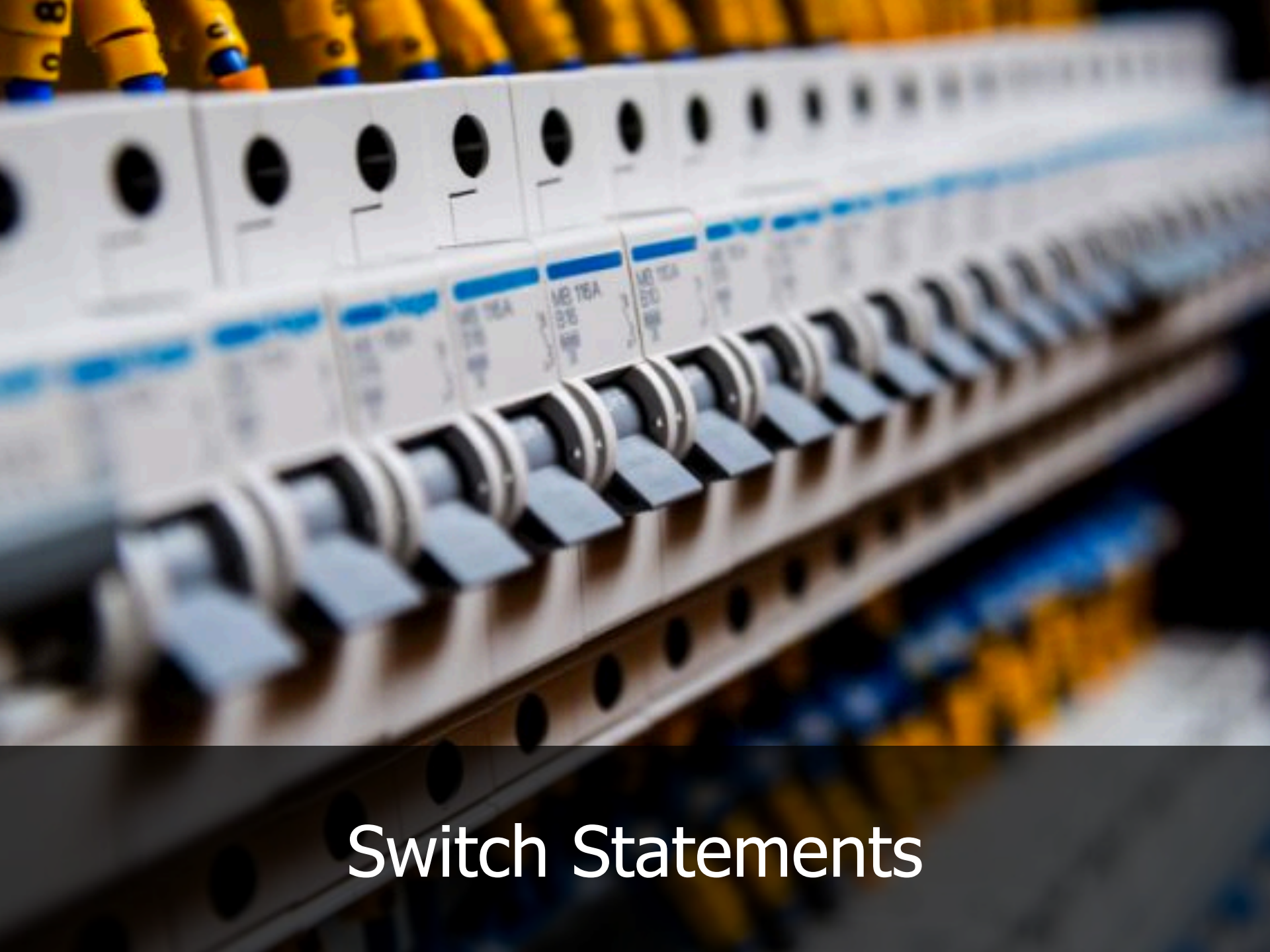
```
1.  function getClass(element) {
2.    if (element.buttonClass) {
3.      return element.buttonClass;
4.    } else {
5.      return "btn-lg";
6.    }
7.  }
```

```
1.      function getClass(element) {
2.        return (element.buttonClass) ? element.buttonClass : "btn-lg";
3.        return element.buttonClass || "btn-lg";
4.      }
```

Remover comando ternário

```
1.  function calculateTax(amount, last12monthRevenue) {
2.    return (last12monthRevenue <= 120000) ? amount * 0.06 :
    ((last12monthRevenue <= 240000) ? amount * 0.08 :
    ((last12monthRevenue <= 360000) ? amount * 0.12 : amount *
    0.15));
3.  }
```

```
1.  function calculateTax(amount, last12monthRevenue) {
2.    if (last12monthRevenue <= 120000) {
3.      return amount * 0.06;
4.    }
5.    if (last12monthRevenue <= 240000) {
6.      return amount * 0.08;
7.    }
8.    if (last12monthRevenue <= 360000) {
9.      return amount * 0.12;
10.   }
11.   return amount * 0.15;
12. }
```



Switch Statements

```
51          this.boleto = boleto;
52
53          if (codBanco == JBoleto.BANCO_DO_BRASIL) {
54
55              banco = new BancoBrasil(boleto);
56          }
57          else if (codBanco == JBoleto.BRADESCO) {
58
59              banco = new Bradesco(boleto);
60          }
61          else if (codBanco == JBoleto.ITAU) {
62
63              banco = new Itau(boleto);
64          }
65          else if (codBanco == JBoleto.BANCO_REAL) {
66
67              banco = new BancoReal(boleto);
68          }
69          else if (codBanco == JBoleto.CAIXA_ECONOMICA) {
70
71              banco = new CaixaEconomica(boleto);
72          }
73          else if (codBanco == JBoleto.UNIBANCO) {
74
75              banco = new Unibanco(boleto);
76          }
77          else if (codBanco == JBoleto.HSBC) {
78
79              banco = new Hsbc(boleto);
80          }
81          else if (codBanco == JBoleto.SANTANDER) {
82
83              banco = new Santander(boleto);
84          }
```

Substituir switch por polimorfismo

```
function calculateEmployeeCost(category, amount) {
  switch(category) {
    case 'CLT': {
      const inss = amount * 0.03;
      const fgts = amount/12;
      const vacations = (amount/3)/12;
      const thirteenFirstSalary = amount/12;
      const vouchers = 1000;
      return amount + inss + fgts + vacations + thirteenFirstSalary + vouchers;
    }
    case 'ESTAGIARIO': {
      const vouchers = 1000;
      return amount + vouchers;
    }
    case 'PJ': {
      return amount;
    }
  }
}
```

```
1.  class Employee {
2.    constructor (amount) {
3.      this.amont = amount;
4.    }
5.
6.    abstract calculateCost();
7.  }
8.
9.  class EmployeeCLT extends Employee {
10.   constructor (amount) {
11.     super(amount);
12.   }
13.
14.   calculateCost() {
15.     const inss = amount * 0.03;
16.     const fgts = amount/12;
17.     const vacations = (amount/3)/12;
18.     const thirteenFirstSalary = amount/12;
19.     const vouchers = 1000;
20.     return amount + inss + fgts + vacations + thirteenFirstSalary + vouchers;
21.   }
```

Métodos longos

```
84          } catch (CTS.PBUserException e) {
85              messageResult = e.toString();
86              System.out.println("----------------------------------------");
87              System.out.println("Erro ao chamar o componente no servidor:");
88              System.out.println("Erro no componente PB: PBUserException. ");
89              System.out.println(messageResult);
90              System.out.println("----------------------------------------");
91
92              bPrint = true;
93          } catch (org.omg.CORBA.TRANSACTION_ROLLEDBACK f) {
94              messageResult = f.toString();
95              System.out.println("----------------------------------------");
96              System.out.println("Erro ao chamar o componente no servidor:");
97              System.out.println("Erro Corba: CORBA.TRANSACTION_ROLLEDBACK.");
98              System.out.println(messageResult);
99              System.out.println("----------------------------------------");
100
101             bPrint = true;
102         } catch (org.omg.CORBA.COMM_FAILURE g) {
103             messageResult = g.toString();
104             System.out.println("----------------------------------------");
105             System.out.println("Erro ao criar o componente no servidor:");
106             System.out.println("Servidor inoperante.");
107             System.out.println(messageResult);
108             System.out.println("----------------------------------------");
109
110             bPrint = true;
111         } catch (org.omg.CORBA.OBJECT_NOT_EXIST h) {
112             messageResult = h.toString();
113             System.out.println("----------------------------------------");
114             System.out.println("Erro ao criar o componente no servidor:");
115             System.out.println("Objeto não existente no servidor.");
116             System.out.println(messageResult);
117             System.out.println("----------------------------------------");
118
119             bPrint = true;
120         } catch (Exception l) {
121             messageResult = l.toString();
122             System.out.println("----------------------------------------");
123             System.out.println("Erro ao criar o componente no servidor:");
124             System.out.println("Objeto não existente no servidor.");
125             System.out.println(messageResult);
126             System.out.println("----------------------------------------");
```

Extrair Método

```
function printReceipt(table, items) {
    console.log("------------------------");
    console.log(table, new Date());
    console.log("------------------------");
    let total = 0;
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
        total += item.price;
    }
    console.log("------------------------");
    const fee = (total * 10)/100;
    const grandTotal = total + fee;
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("------------------------");
}
```

```
function printReceipt(table, items) {
    console.log("-----------------------");
    console.log(table, new Date());
    console.log("-----------------------");
    let total = 0;
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
        total += item.price;
    }
    console.log("-----------------------");
    const fee = (total * 10)/100;
    const grandTotal = total + fee;
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("-----------------------");
}
```

```
function printHeader(table) {
    console.log("------------------------");
    console.log(table, new Date());
    console.log("------------------------");
}

function printReceipt(table, items) {
    printHeader(table);
    let total = 0;
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
        total += item.price;
    }
    console.log("------------------------");
    const fee = (total * 10)/100;
    const grandTotal = total + fee;
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("------------------------");
}
```

```
function printHeader(table) {
    console.log("------------------------");
    console.log(table, new Date());
    console.log("------------------------");
}

function printReceipt(table, items) {
    printHeader(table);
    let total = 0;
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
        total += item.price;
    }
    console.log("------------------------");
    const fee = (total * 10)/100;
    const grandTotal = total + fee;
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("------------------------");
}
```

```
function printHeader(table) {
    console.log("------------------------");
    console.log(table, new Date());
    console.log("------------------------");
}
function calculateTotal(items) {
    let total = 0;
    for (const item of items) {
        total += item.price;
    }
    return total;
}
function printReceipt(table, items) {
    printHeader(table);
    let total = calculateTotal(items)
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
    }
    console.log("------------------------");
    const fee = (total * 10)/100;
    const grandTotal = total + fee;
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("------------------------");
}
```

```
function printHeader(table) {
    console.log("------------------------");
    console.log(table, new Date());
    console.log("------------------------");
}
function calculateTotal(items) {
    let total = 0;
    for (const item of items) {
        total += item.price;
    }
    return total;
}
function printReceipt(table, items) {
    printHeader(table);
    let total = calculateTotal(items)
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
    }
    console.log("------------------------");
    const fee = (total * 10)/100;
    const grandTotal = total + fee;
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("------------------------");
}
```

```
function printHeader(table) {
    console.log("------------------------");
    console.log(table, new Date());
    console.log("------------------------");
}
function calculateTotal(items) {
    let total = 0;
    for (const item of items) {
        total += item.price;
    }
    return total;
}
function printItems(items) {
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
    }
}
function printReceipt(table, items) {
    printHeader(table);
    let total = calculateTotal(items)
    printItems(items);
    console.log("------------------------");
    const fee = (total * 10)/100;
    const grandTotal = total + fee;
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("------------------------");
}
```

```
function printHeader(table) {
    console.log("------------------------");
    console.log(table, new Date());
    console.log("------------------------");
}
function calculateTotal(items) {
    let total = 0;
    for (const item of items) {
        total += item.price;
    }
    return total;
}
function printItems(items) {
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
    }
}
function printReceipt(table, items) {
    printHeader(table);
    let total = calculateTotal(items)
    printItems(items);
    console.log("------------------------");
    const fee = (total * 10)/100;
    const grandTotal = total + fee;
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("------------------------");
}
```

```
function printHeader(table) {
    console.log("------------------------");
    console.log(table, new Date());
    console.log("------------------------");
}
function calculateTotal(items) {
    let total = 0;
    for (const item of items) {
        total += item.price;
    }
    return total;
}
function printItems(items) {
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
    }
}
function calculateFee(total) {
    return (total * 10)/100;
}
function printReceipt(table, items) {
    printHeader(table);
    let total = calculateTotal(items)
    printItems(items);
    console.log("------------------------");
    const fee = calculateFee(total);
    const grandTotal = total + fee;
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("------------------------");
}
```

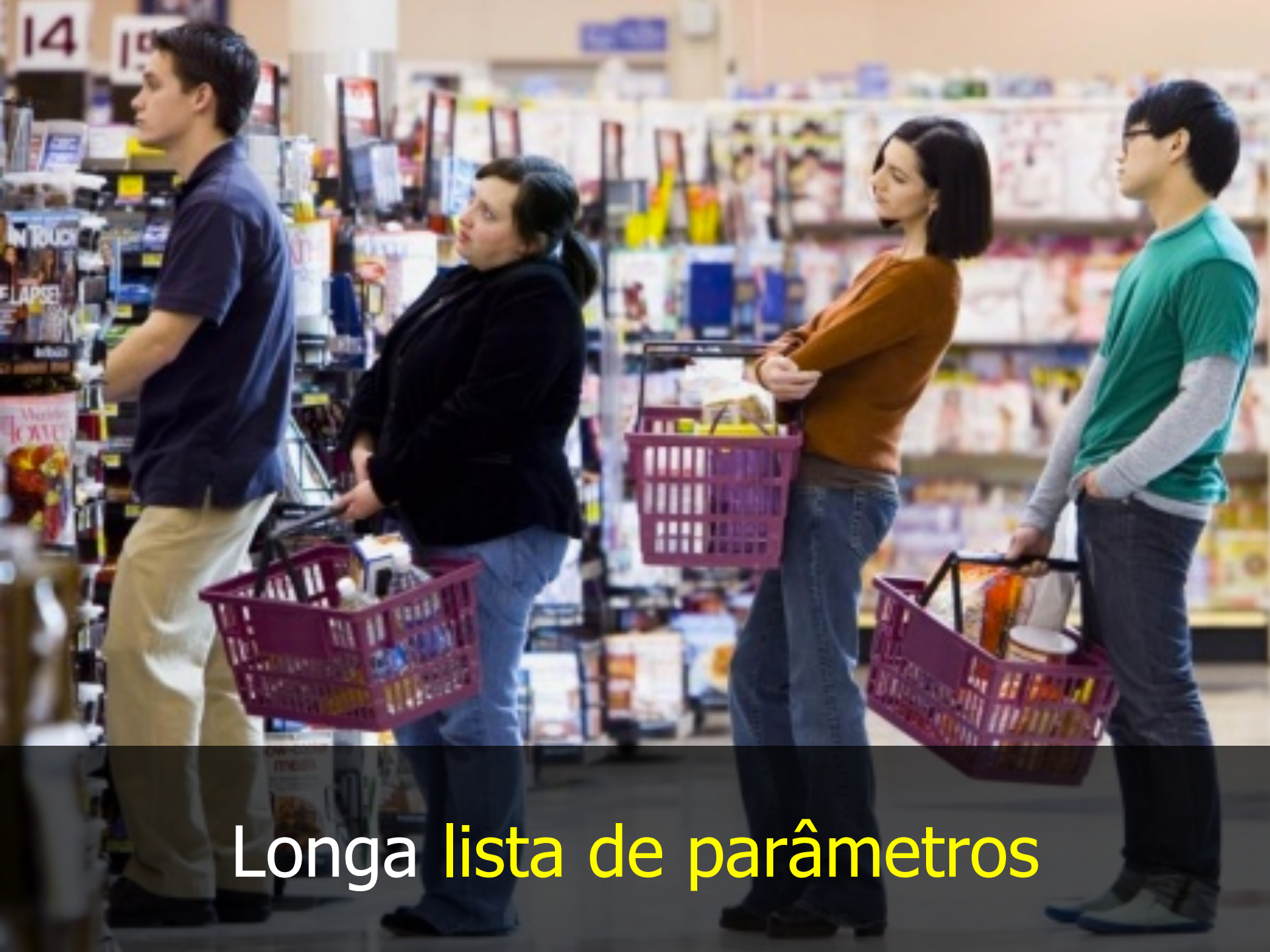
```
function printHeader(table) {
    console.log("------------------------");
    console.log(table, new Date());
    console.log("------------------------");
}
function calculateTotal(items) {
    let total = 0;
    for (const item of items) {
        total += item.price;
    }
    return total;
}
function printItems(items) {
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
    }
}
function calculateFee(total) {
    return (total * 10)/100;
}
function printReceipt(table, items) {
    printHeader(table);
    let total = calculateTotal(items)
    printItems(items);
    console.log("------------------------");
    const fee = calculateFee(total);
    const grandTotal = total + fee;
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("------------------------");
}
```

```
function printHeader(table) {
    console.log("------------------------");
    console.log(table, new Date());
    console.log("------------------------");
}
function calculateTotal(items) {
    let total = 0;
    for (const item of items) {
        total += item.price;
    }
    return total;
}
function printItems(items) {
    for (const item of items) {
        console.log(`${item.description} ${item.price}`);
    }
}
function calculateFee(total) {
    return (total * 10)/100;
}
function printTotal(total, fee, grandTotal) {
    console.log(`Total: ${total}`);
    console.log(`Fee: ${fee}`);
    console.log(`Grand total: ${grandTotal}`);
    console.log("------------------------");
}
function printReceipt(table, items) {
    printHeader(table);
    let total = calculateTotal(items)
    printItems(items);
    console.log("------------------------");
    const fee = calculateFee(total);
    const grandTotal = total + fee;
    printTotal(total, fee, grandTotal);
}
```

Internalizar Método

```
1.    function evaluatePoints( ) {
2.      return (moreThan5Delays()) ? 2 : 1;
3.    }
4.
5.    function moreThan5Delays() {
6.      return numberOfDelays > 5;
7.    }
```

```
1.  function evaluatePoints( ) {
2.    return (numberOfDelays > 5) ? 2 : 1;
3.  }
```


Longa lista de parâmetros

Preservar o objeto inteiro

```
1.  function getBalance(dueDate, amount, penaltyPercentage, interestPercentage) {
2.    const dateDiffInMilli = new Date().getTime() - dueDate.getTime();
3.    const milliToDayFactor = 1000*60*60*24;
4.    const dateDiffInDays = Math.floor(dateDiffInMilli/milliToDayFactor);
5.    if (dateDiffInDays <= 0) return amount;
6.    const penaltyAmount = (amount*penaltyPercentage)/100;
7.    const interestAmount = ((amount*interestPercentage)/100)*dateDiffInDays
8.    return amount + penaltyAmount + interestAmount;
9.  }
```

```
1.  function getBalance(invoice) {
2.    const dateDiffInMilli = new Date().getTime() - invoice.dueDate.getTime();
3.    const milliToDayFactor = 1000*60*60*24;
4.    const dateDiffInDays = Math.floor(dateDiffInMilli/milliToDayFactor);
5.    if (dateDiffInDays <= 0) return invoice.amount;
6.    const penaltyAmount = (invoice.amount*invoice.penaltyPercentage)/100;
7.    const interestAmount = ((invoice.amount*invoice.interestPercentage)/
    100)*dateDiffInDays
8.    return invoice.amount + penaltyAmount + interestAmount;
9.  }
```

Introduzir objeto parâmetro

```
1.  function calculateGrade(studentAnswers, correctAnswers, startDate, endDate) {
2.    let total = 0;
3.    for (const answer of studentAnswers) {
4.      if (answer.value === correctAnswers[answer.id]) {
5.        total++;
6.      }
7.    }
8.    const milliToMinutesFactor = 1000*60;
9.    const duration = (endDate.getTime() - startDate.getTime())/milliToMinutesFactor;
10.   if (duration < 60) {
11.     return total + (total*0.10);
12.   } else {
13.     return total;
14.   }
15. }
```

```
1.  function calculateGrade(studentAnswers, correctAnswers, period) {
2.    let total = 0;
3.    for (const answer of studentAnswers) {
4.      if (answer.value === correctAnswers[answer.id]) {
5.        total++;
6.      }
7.    }
8.    if (duration < period.getTimeInMinutes()) {
9.      return total + (total*0.10);
10.   } else {
11.     return total;
12.   }
13. }
```

Extrair classe

```
1.  function getBalance(invoice) {
2.    const dateDiffInMilli = new Date().getTime() - invoice.dueDate.getTime();
3.    const milliToDayFactor = 1000*60*60*24;
4.    const dateDiffInDays = Math.floor(dateDiffInMilli/milliToDayFactor);
5.    if (dateDiffInDays <= 0) return invoice.amount;
6.    const penaltyAmount = (invoice.amount*invoice.penaltyPercentage)/100;
7.    const interestAmount = ((invoice.amount*invoice.interestPercentage)/
    100)*dateDiffInDays
8.    return invoice.amount + penaltyAmount + interestAmount;
9.  }
```

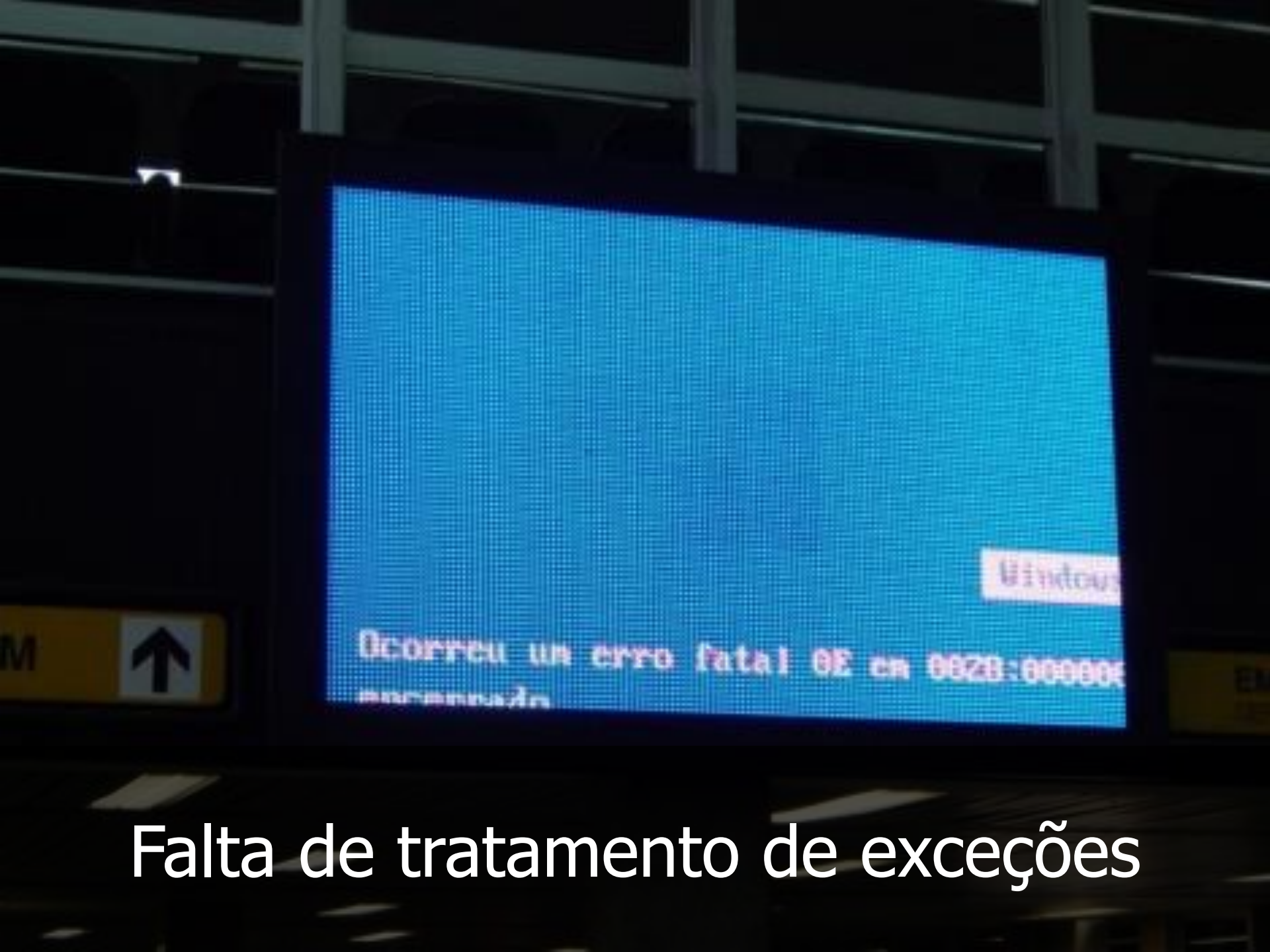
```
1.  class Invoice {
2.    constructor (dueDate, amount, penaltyPercentage, interestPercentage) {
3.      this.dueDate = dueDate;
4.      this.amount = amount;
5.      this.penaltyPercentage = penaltyPercentage;
6.      this.interestPercentage = interestPercentage;
7.    }

9.    getBalance() {
10.     const dateDiffInMilli = new Date().getTime() - this.dueDate.getTime();
11.     const milliToDayFactor = 1000*60*60*24;
12.     const dateDiffInDays = Math.floor(dateDiffInMilli/milliToDayFactor);
13.     if (dateDiffInDays <= 0) return this.amount;
14.     const penaltyAmount = (this.amount*this.penaltyPercentage)/100;
15.     const interestAmount = ((this.amount*this.interestPercentage)/
    100)*dateDiffInDays
16.     return this.amount + penaltyAmount + interestAmount;
17.   }
18. }
```

Remover atribuições a parâmetros

```
1.    function applyCoupon(cart, coupon) {
2.      let total = 0;
3.      for (const item of cart) {
4.        total += item.price;
5.      }
6.      cart.discount = (total*coupon.percentage)/100;
7.    }
```

```
1.      function calculateDiscount(cart, coupon) {
2.        let total = 0;
3.        for (const item of cart) {
4.          total += item.price;
5.        }
6.        return (total*coupon.percentage)/100;
7.      }
```



Falta de tratamento de exceções

```
682                         if (teste1 != teste2){
683                             break;
684                         }
685                     }
686                     meses.add(listDias);
687                 }
688             }else {
689                 MessageUtil.actionMessages(messages, result.bufferMessage);
690             }
691             }catch (Exception e) {
692                 e.printStackTrace();
693             }
694             return meses;
695     }
696
```

Tratar exceções de forma adequada

```
1.  function createTransaction(token, transaction) {
2.    try {
3.      axios({
4.        url: "https://api.paypal.com/transaction",
5.        method: "post",
6.        headers: {
7.          Authentication: `Bearer ${token}`
8.        },
9.        data: paypalAdapter.adapt(transaction)
10.     });
11.   } catch (e) {
12.   }
13. }
```

```
1.  function createTransaction(token, transaction) {
2.    try {
3.      axios({
4.        url: "https://api.paypal.com/transaction",
5.        method: "post",
6.        headers: {
7.          Authentication: `Bearer ${token}`
8.        },
9.        data: paypalAdapter.adapt(transaction)
10.     });
11.   } catch (e) {
12.     logRepository.saveLog(e);
13.   }
14. }
```

Lançar exceções com informações

```
1.  function createTransaction(token, transaction) {
2.    try {
3.      axios({
4.        url: "https://api.paypal.com/transaction",
5.        method: "post",
6.        headers: {
7.          Authentication: `Bearer ${token}`
8.        },
9.        data: paypalAdapter.adapt(transaction)
10.     });
11.   } catch (e) {
12.     logRepository.saveLog(e);
13.     throw new Error("Transaction error");
14.   }
15. }
```

```
1.  function createTransaction(token, transaction) {
2.    try {
3.      axios({
4.        url: "https://api.paypal.com/transaction",
5.        method: "post",
6.        headers: {
7.          Authentication: `Bearer ${token}`
8.        },
9.        data: paypalAdapter.adapt(transaction)
10.     });
11.   } catch (e) {
12.     logRepository.saveLog(e);
13.     throw new TransactionError(transaction, e);
14.   }
15. }
```

Substituir tratamento de exceção por condição

```
1.  public class ConnectionPool {
2.    private Stack available;
3.
4.    public Connection getConnection() {
5.      Connection connection;
6.      try {
7.        connection = (Connection) available.pop();
8.        return connection;
9.      } catch(EmptyStackException e) {
10.       connection = new Connection();
11.       return connection;
12.     }
13.   }
14. }
```

```
1.  public class ConnectionPool {
2.    private Stack available;
3.
4.    public Connection getConnection() {
5.      Connection connection;
6.      if (available.isEmpty()) {
7.        connection = new Connection();
8.        return connection;
9.      }
10.     connection = (Connection) available.pop();
11.     return connection;
12.   }
13. }
```

```
1.  public class ConnectionPool {
2.    private Stack available;
3.
4.    public Connection getConnection() {
5.      Connection connection;
6.      if (available.isEmpty()) {
7.        return new Connection();
8.      }
9.      connection = (Connection) available.pop();
10.     return connection;
11.   }
12. }
```

```
1.  public class ConnectionPool {
2.    private Stack available;
3.
4.    public Connection getConnection() {
5.      Connection connection;
6.      if (available.isEmpty()) {
7.        return new Connection();
8.      }
9.      return (Connection) available.pop();
10.   }
11. }
```

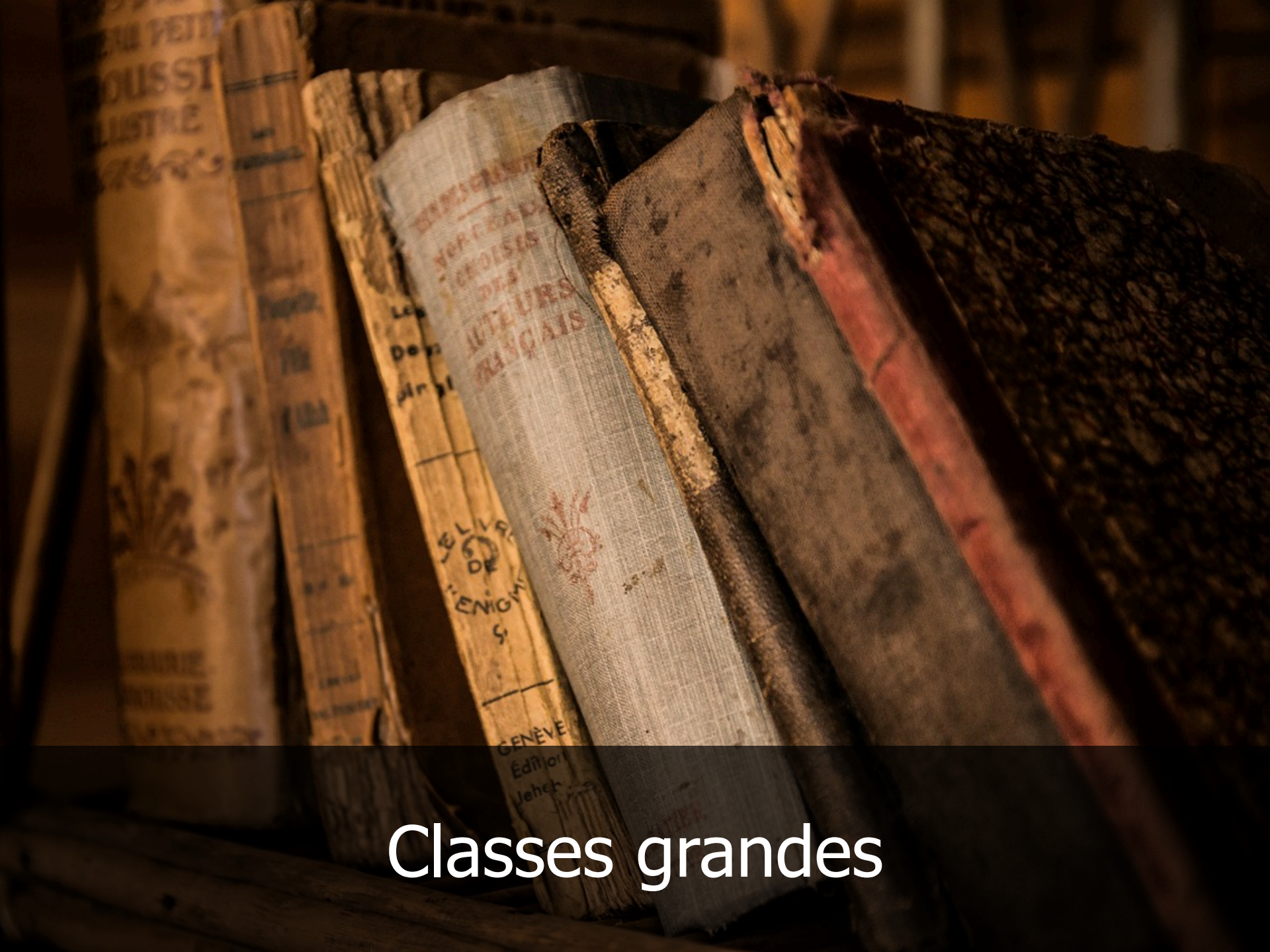
```
1.      public class ConnectionPool {
2.        private Stack available;
3.
4.        public Connection getConnection() {
5.          if (available.isEmpty()) {
6.            return new Connection();
7.          }
8.          return (Connection) available.pop();
9.        }
10.     }
```

```
1.  public class ConnectionPool {
2.    private Stack available;
3.
4.    public Connection getConnection() {
5.      if (available.isEmpty()) return new Connection();
6.      return (Connection) available.pop();
7.    }
8.  }
```

Relançar exceções adequadas ao domínio

```
1.  class RepositoryDatabase {
2.
3.    getCustomer(id) {
4.      // db.one lança um erro "No data returned from query"
5.      return db.one("select * from customer where id = $1", [id]);
6.    }
7.  }
```

```
1.  class RepositoryDatabase {
2.
3.    getCustomer(id) {
4.      const customer = await db.oneOrNone("select * from customer where id = $1", [id]);
5.      if (!customer) {
6.        throw new CustomerNotFound(id);
7.      }
8.    }
9.  }
```



Classes grandes

Extrair classe

```
1.  class Invoice {
2.    constructor (dueDate, amount, penaltyPercentage, interestPercentage) {
3.      this.dueDate = dueDate;
4.      this.amount = amount;
5.      this.penaltyPercentage = penaltyPercentage;
6.      this.interestPercentage = interestPercentage;
7.    }
8.
9.    getBalance() {
10.     const dateDiffInMilli = new Date().getTime() - this.dueDate.getTime();
11.     const milliToDayFactor = 1000*60*60*24;
12.     const dateDiffInDays = Math.floor(dateDiffInMilli/milliToDayFactor);
13.     if (dateDiffInDays <= 0) return this.amount;
14.     const penaltyAmount = (this.amount*this.penaltyPercentage)/100;
15.     const interestAmount = ((this.amount*this.interestPercentage)/
    100)*dateDiffInDays
16.     return this.amount + penaltyAmount + interestAmount;
17.   }
18. }
```

```
1.  class Invoice {
2.    constructor (dueDate, amount, penaltyPercentage, interestPercentage) {
3.      this.dueDate = dueDate;
4.      this.amount = amount;
5.      this.penaltyPercentage = penaltyPercentage;
6.      this.interestPercentage = interestPercentage;
7.    }
8.
9.    getDiffInDays() {
10.     const dateDiffInMilli = new Date().getTime() - this.dueDate.getTime();
11.     const milliToDayFactor = 1000*60*60*24;
12.     return Math.floor(dateDiffInMilli/milliToDayFactor);
13.   }
14.
15.   getBalance() {
16.     const dateDiffInDays = getDiffInDays();
17.     if (dateDiffInDays <= 0) return this.amount;
18.     const penaltyAmount = (this.amount*this.penaltyPercentage)/100;
19.     const interestAmount = ((this.amount*this.interestPercentage)/
    100)*dateDiffInDays
20.     return this.amount + penaltyAmount + interestAmount;
21.   }
22. }
```

```
1.  class Period {
2.    constructor (startDate, endDate) {
3.      this.startDate = startDate;
4.      this.endDate = endDate;
5.    }
6.
7.    getDiffInDays() {
8.      const dateDiffInMilli = this.startDate.getTime() - this.endDate.getTime();
9.      const milliToDayFactor = 1000*60*60*24;
10.     return Math.floor(dateDiffInMilli/milliToDayFactor);
11.   }
12. }
```

```
1.  class Invoice {
2.    constructor (dueDate, amount, penaltyPercentage, interestPercentage) {
3.      this.dueDate = dueDate;
4.      this.amount = amount;
5.      this.penaltyPercentage = penaltyPercentage;
6.      this.interestPercentage = interestPercentage;
7.    }
8.
9.    getDiffInDays() {
10.     const period = new Period(new Date(), this.dueDate);
11.     return period.getDiffInDays();
12.   }
13.
14.   getBalance() {
15.     const dateDiffInDays = getDiffInDays();
16.     if (dateDiffInDays <= 0) return this.amount;
17.     const penaltyAmount = (this.amount*this.penaltyPercentage)/100;
18.     const interestAmount = ((this.amount*this.interestPercentage)/
    100)*dateDiffInDays
19.     return this.amount + penaltyAmount + interestAmount;
20.   }
21. }
```

```
1.  class Invoice {
2.    constructor (dueDate, amount, penaltyPercentage, interestPercentage) {
3.      this.dueDate = dueDate;
4.      this.amount = amount;
5.      this.penaltyPercentage = penaltyPercentage;
6.      this.interestPercentage = interestPercentage;
7.    }
8.
9.    getBalance() {
10.     const period = new Period(new Date(), this.dueDate);
11.     const dateDiffInDays = period.getDiffInDays();
12.     if (dateDiffInDays <= 0) return this.amount;
13.     const penaltyAmount = (this.amount*this.penaltyPercentage)/100;
14.     const interestAmount = ((this.amount*this.interestPercentage)/
    100)*dateDiffInDays
15.     return this.amount + penaltyAmount + interestAmount;
16.   }
17. }
```