funções

Esses exercicios foca em Erros do compilador e de execução.

O objetivo desses exercicis é:

- Escrever os códigos
- Ler o erro que o compilador ou o ambiente de execução devolve
- Escrever o código correto para que, ao executá-lo, não aparecer mais erros.

Importante conhecer um pouco sobre jshell, comandos importantes para esses exercícios:

Para chamar um método/função:

```
<nome do método>()
```

Salvar uma sessão do jshell:

```
/save <nome>
```

Para uma sessão salva do jshell

```
/open <nome>
```

Dentro de um recurso, se quiser listar os metodos/funções salvas:

```
/list
```

Sair do jshell:

```
/exit
```

E1.1

```
err1() {
return "¿Dónde está el error?";
}
```

Erro:

```
|  Error:
|  ';' expected
|  err1() {
|         ^
|  Error:
|  cannot find symbol
|    symbol:   method err1()
|  err1() {
|  ^--^
```

“Não é possível encontrar o símbolo”

está faltando o tipo de retorno da função

Correção:

```
String err1() {
   return "¿Dónde está el error?";
}
```

Exercicio salvo:

```
jshell> /save e1.1

jshell> /list

   1 : String err1() {
           return "¿Dónde está el error?";
       }

jshell> err1()
$2 ⟹ "¿Dónde está el error?"
```

E1.2

```
string err2() {
    return "¿String o string?";
}
```

Erro:

```
 created method err2(), however, it cannot be referenced until class string i
s declared
```

“ criou o método err2(), no entanto, ele não pode ser referenciado até que a classe string seja declarada”,

É um erro comum de escrita do tipo do retorno, assim gerando um erro léxico.

Correção:

```
String err2() {
   return "¿String o string?";
}
```

Exercicio salvo:

```
jshell> /save e1.2

jshell> /exit

jshell> /open e1.2

jshell> /list

   1 : String err2() {
           return "¿String o string?";
       }
   2 : err2()

jshell> /open e1.2
```

```
jshell> err2()
$5 ==> "¿String o string?"
```

E1.3

```
String err3(who) {
    return String.format("%s, cuál es el error?", who);
}
```

Erro:

```
|  Error:
|  <identifier> expected
|  String err3(who) {
|                  ^
```

“Identificador esperado”

Faltou especificar o tipo do parametro da função, que é uma String.

Correção:

```
String err3(String who) {
    return String.format("%s, cuál es el error?", who);
}
```

Exercicio salvo:

```
jshell> /save err3

jshell> /exit
|  Goodbye

jshell> /open err3

jshell> /list

   1 : String err3(String who) {
```

```
        return String.format("%s, cuál es el error?", who);
    }
  2 : err3("meu erro em PT-BR")

jshell> err3("novamente")
$3 ⟹ "novamente, cuál es el error?"
```

E1.4

```
String err4() {
   return "Este es un error sutil"
}
```

Erro:

```
|  Error:
|  ';' expected
|     return "Este es un error sutil"
|                                   ^
```

“esperava “;””

Em todo comando/sentença em Java, deve finalizar com ponto e vírgula.

Correção:

```
String err4() {
   return "Este es un error sutil";
}
```

Bom acho que deu pra reparar que já sei salvar um recurso do jshell, criar ele e listar as funções criadas, não irei mais repetir.

E1.5

```
String err5() {
"Es es un poco menos sutil";
```

```
}
```

Erro:

```
| Error:
| not a statement
|     "Es es un poco menos sutil";
|     ^------------------------^
| Error:
| missing return statement
| String err5() {
|               ^
```

“não é uma declaração, declaração de retorno ausente”

Está faltando o return.

Correção:

```
String err5() {
    return "Es es un poco menos sutil";
}
```

E1.6

```
String err6 {
    return "¡Hola Error!";
}
```

Erro:

```
| Error:
| ';' expected
| String err6 {
|             ^
```

“esperava “;””

Porém não necessariamente é um erro de falta de ponto e vírgula, na verdade o erro está bem próximo, está faltando o selamento da função err6, quando criamos uma função, devemos explicitar o parenteses, mesmo que não há argumentos para a função.

Correção:

```
String err6() {
   return "¡Hola Error!";
}
```

E1.7

```
String err7(String quien) {
   return format("%s ¿Un poco mañoso, no?", quien);
}
```

Erro:

```
| modified method err7(String), however, it cannot be invoked until method
format(java.lang.String,java.lang.String) is declared
```

“O método modificado err7(String), no entanto, não pode ser chamado até que o método format(java.lang.String,java.lang.String) seja declarado”

Está faltando a declaração “String” antes do format.

Correção:

```
String err7(String quien) {
   return String.format("%s ¿Un poco mañoso, no?", quien);
}
```

E1.8

```
String _A_(String emoji1_B_ String emoji2_C_ String emoji3) _D_
return _E_("[%s, %s, %s]"_F_ emoji1_G_ emoji2_H_ emoji3);
```

```
_I_
```

1. ,
2. ,
3. ,
4. ,
5. ,
6. }
7. {
8. adivinaLaPelicula
9. String.format

Correção: irei utilizar respectivamente as opções: 8, 1, 2, 7, 3, 4, 5, 6

```
String adivinaLaPelicula(String emoji1, String emoji2, String emoji3) {
    return String.format("[%s, %s, %s]", emoji1, emoji2, emoji3);
}
```

E1.8

```
_A_ _B_ {
_C_ String.format("こんにちは_D_", _E_)_F_
}
```

1. return
2. ;
3. dare
4. konnichiwa(String dare)
5. %s
6. String

Correção: irei utilizar respectivamente as opções: 6, 4, 1, 5, 3, 2

```
String konnichiwa(String dare) {
    return String.format("こんにちは_%s", dare);
}
```

E1.9

Função separador criada:

```
String separador() {
    return "<<<>>>";
}
```

E1.10

Função vocales criada:

```
String vocales() {
    return "aeiou";
}
```

E1.11

Função tituloAdornado criada:

```
String tituloAdornado(String titulo) {
    return String.format("<<< %s >>>", titulo);
}
```

E1.12

Função dada:

```
String adornar(String frase) {
    return String.format("<=%s⇒, frase");
}
```

Função adornar corrigida:

```
String adornar(String frase) {
   return String.format("<=%s⇒", frase);
}
```

E1.13

Função dada:

```
String concatenar(String s1, String s2) {
   return String.format("[%s+%s]", s2, s1);
}
```

Função concatenar corrigida:

```
String concatenar(String s1, String s2) {
   return String.format("[%s+%s]", s1, s2);
}
```