

Linguagens e Ambientes de Execução

Exercícios #1, Verão de 2022/2023

Dificuldade: ●○○○○

1. [2.5] Considere o programa Kotlin do ficheiro fonte indicado ao lado. Indique os nomes dos ficheiros `.class` resultantes da sua compilação.

```
data class Message(val txt : String)
class Printer {
    fun writeln(msg : Message) { println(msg.txt) }
}
val out = Printer()
fun main() {
    out.writeln(Message("Hello"))
}
```

prog.kt

2. [2.5] Indique as propriedades que podem ser encontradas no tipo `Foo` consultando `memberProperties`

```
class Foo(a: Int, var b: String, val c: Int, d: Double)
{
    val x : String = "ISEL"
    var y : Int = 2023
}
```

3. [3] Escreva, em Java, o método estático `setMany`, que atribui o valor `value` a uma propriedade dos objectos indicados em `objs`, usando o método `setter` indicado em `propSetter`.

```
public static <T, V> void setMany(T[] objs, Method propSetter, V value)
```

4. [3] Apresente uma função em Kotlin equivalente à descrição em *bytecode* Java apresentada ao lado.

```
public static final int func(int, int);
Code:
0: iload_0
1: iload_1
2: imul
3: iconst_5
4: iadd
5: ireturn
```

5. [3] Considere a função de extensão `weaklyCheckedAs` para `List<*>` :

```
1 fun <R> List<*>.weaklyCheckedAs() : List<R> {
2     if (!isEmpty()) {
3         first() as R
4     }
5     @Suppress("UNCHECKED_CAST")
6     return this as List<R>
7 }
```

- a. [1.5] Para que o `cast` da linha 3 seja efetivo, a linha 1 precisa de duas alterações. Sem essas alterações, é emitido um aviso de *“unchecked cast”* na linha 3 e o `cast` é inútil. Apresente e justifique a correção.
NOTA: a correção **não** passa por usar a anotação `@Suppress` para ocultar o aviso.
- b. [1.5] A linha 5 oculta um aviso de `cast` não verificado na linha 6. Explique porque o `cast` não é verificável.

6. [2.5] Recorrendo a `sequence` e `yield`, construa a função de extensão `noNulls` para `Sequence<T?>`, que produz de forma *lazy* uma sequência com os mesmos elementos da sequência de entrada, excepto os `null`.

```
fun <T : Any> Sequence<T?>.noNulls(): Sequence<T>
```

7. [2] Considerando um esquema de recolha automática de memória (*garbage collection*) baseado em marcação (*mark/marking*), descreva de forma sucinta como se determina quais os objetos que podem ser recolhidos.
8. [1.5] Qual das seguintes implementações mais se aproxima da função de extensão `use` definida na biblioteca padrão da linguagem Kotlin? Justifique a sua escolha.

<pre>fun <T : Closeable, R> T.use(func: (T) -> R) : R { try { return func(this) } finally { close() } }</pre> <div>A</div>	<pre>fun <T> Collection<T>.use(func: (T) -> Unit) { for (item in this) { func(item) } }</pre> <div>B</div>	<pre>fun <T : Closeable, R> Sequence<T>.use(func: (T) -> R) : Sequence<R> = sequence { for (item in this@use) { try { yield(func(this)) } finally { item.close() } } }</pre> <div>C</div>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Duração: ilimitada
ISEL, 26 de maio de 2023