Instituto Superior de Engenharia de Lisboa

LEIC

Técnicas de Desenvolvimento de Software

Teste Final (Época Normal), Verão de 2021/2022, 24 de Junho 2022

Nome: Número:

Grupo I

Considerando os temas tratados nas aulas da unidade curricular, responda às perguntas seguintes assinalando de forma inequívoca a opção correta. Não responda arbitrariamente: cada resposta incorreta desconta 1/3 da cotação da pergunta ao total obtido no grupo. A cotação total do grupo é 7 valores e todas as perguntas têm a mesma cotação.

1. Qual das seguintes funções provocaria erro de compilação

   - ☐ `fun String.f1(): Unit = println(this)`
   - ☐ `fun f2(): Nothing = throw IllegalStateException()`
   - ☐ `fun <T> f3(arg: T?): Any = (arg != null)`
   - ☐ `fun Int?.f4(): Any = this`

2. Qual é a sequência de dígitos apresentados pela execução do seguinte troço de código:

   - ☐ **1234567**
   - ☐ **1725346**
   - ☐ **125367**
   - ☐ **172536**
   - ☐ **12357**
   - ☐ **17253**

```
print(1)
runBlocking {
    print(2)
    val job = launch {
        print(3); delay(100); print(4)
    }
    print(5); delay(50); job.cancel(); print(6)
}
print(7)
```

3. Identifique a frase correta quanto à utilização dos modificadores **`abstract`**, **`open`** e **`override`** na herança.

   - ☐ Os três modificadores podem ser aplicados em classes e em funções internas (métodos).
   - ☐ As classes marcadas com **`open`** não são abstratas mas podem ter métodos marcados com **`abstract`**.
   - ☐ Qualquer classe deriva explicitamente de uma classe marcada com **`open`** ou **`abstract`**, ou deriva implicitamente da classe **`Any`**, que está marcada com **`open`**.
   - ☐ As classes com métodos marcados com **`override`** derivam obrigatoriamente de classes marcadas com **`open`** em que esses métodos estão marcados com **`abstract`**.

4. Identifique a frase incorreta quanto à chamada **`value.f()`** sendo **`f`** uma função interna (método) de uma classe ou uma função extensão dessa classe.

   - ☐ A chamada pode ser polimórfica se **`f`** for interna, mas não é polimórfica se for extensão.
   - ☐ **`f`** tem acesso às propriedades privadas da classe se for interna, mas não tem acesso se for extensão.
   - ☐ Como interna, **`f`** pode ser redefinida numa classe derivada, mas como extensão não pode haver outra função **`f`** que seja extensão de uma classe derivada.
   - ☐ Não faz sentido definir **`f`** como interna numa classe e definir também **`f`** como extensão dessa classe.

Considerando as seguintes declarações responda aos pontos 5 e 6

```
sealed class A(val a: Char='A')
class B(val x:Int): A('B')
object C: A('C') { val y = 20 }

fun f(a: A) = when(a) {
    is B -> a.x
    is C -> a.y
}
```

```
fun main() {
    val l1 = listOf( B(10), C, B(30) )
    val l2 = l1.map { "${it.a}${f(it)}" }
    println(l2)
}
```

5. Identifique a frase incorreta

- [ ] O tipo de retorno implícito da função **`f`** é **`Int`**
- [ ] O **`when`** da função **`f`** é exaustivo porque a hierarquia é fechada
- [ ] O tipo implícito de **`l2`** é **`List<String>`**
- [ ] A classe **`A`** poderia ser marcada com **`abstract`** em vez de **`sealed`**

6. Identifique a frase incorreta

- [ ] O texto apresentado pela chamada ao ***`println`*****`(l2)`** é: **`[B10, C20, B30]`**
- [ ] Cada ramo do **`when`** faz um *smart cast* de **`a`** para o tipo concreto
- [ ] É desnecessário o valor por omissão do construtor primário da classe **`A`**
- [ ] O tipo implícito de **`l1`** é **`List<Any>`**

Considerando as seguintes declarações responda ao ponto 7

```
object X: Closeable {
    init { print("Begin;") }
    fun action() { print("Action;") }
    override fun close() { print("End;") }
}
```

```
fun main() {
    X.action()
    X.use{ it.action() }
}
```

7. O *output* produzido pela execução da função **`main`** é:

- [ ] **`Begin;Action;Action;End;`**
- [ ] **`Action;Begin;Action;End;`**
- [ ] **`Begin;Action;Begin;Action;End;`**
- [ ] Nenhuma das hipóteses anteriores.

Grupo II

Pretende-se desenvolver uma aplicação em *Desktop Compose* para converter notas quantitativas (de 0 a 20 valores) em notas qualitativas (Suficiente, Bom, etc.). Para tal, e com o objetivo de não permitir a representação de estados inválidos (*Make Illegal States Unrepresentable*), foram declarados os tipos **`Grade`** e **`QualitativeGrade`**.

```
class Grade private constructor(val value: Int) {
   companion object { val scale = (0..20).map { Grade(it) } }
}
enum class QualitativeGrade(val range: IntRange, val description: String) {
   MB(18..20, "Muito Bom"), B(14..17, "Bom"), S(10..13, "Suficiente"),
   I(6..9, "Insuficiente"), M(0..5, "Mau")
}
```

A figura ao lado apresenta a janela da aplicação, quando é iniciada. Na parte inferior aparece a nota qualitativa correspondente à nota selecionada na parte superior.

A função `main` da aplicação é a seguinte.

```
fun main() = application {
   val state = WindowState( size = DpSize(600.dp, Dp.Unspecified) )
   Window(onCloseRequest = ::exitApplication, state = state, title = "Grades Converter") {
      var converter by remember { mutableStateOf(GradeConverter(10.grade)) }
      Column(Modifier.padding(5.dp), horizontalAlignment = Alignment.CenterHorizontally) {
         GradeSelector(converter.quantitative) { sel: Grade ->
            converter = GradeConverter(sel)
         }
         QualitativeView(converter.qualitative)
      }
   }
}
```

1. [1] Implemente a função extensão `Int.toGradeOrNull():Grade?` , que converte um valor inteiro numa nota ou retorna `null` se o valor não pertencer à escala. Use apenas a informação disponível no tipo **`Grade`**.

2. [1] Usando apenas a informação disponível no tipo **`QualitativeGrade`**, implemente a função extensão de conversão `Grade.toQualitative():QualitativeGrade`.

3. [1] Chamando a função do ponto 1, defina a propriedade extensão ***`grade`*** usada, por exemplo, em **`10.grade`**, para converter um valor do tipo `Int` em `Grade`.

4. [2] Defina o tipo **imutável `GradeConverter`**, tendo em atenção a sua utilização na função `main`.

5. [2] Crie dois testes automáticos para verificação do tipo **`GradeConverter`**. Um teste para uma utilização válida e outro para uma utilização inválida.

6. [3] Implemente o *Composable* `GradeSelector`, usado na função `main`, com o aspecto aproximado ao da figura. Este componente deve ajustar-se à largura da janela e cada `RadioButton` deve ter 25*dp*. Este componente recebe dois parâmetros: a nota selecionada e a função a chamar quando selecionada uma nota.

7. [1] Implemente o *Composable* `QualitativeView`, que apresenta a descrição da nota qualitativa indicada como parâmetro.

8. [2] Descreva todas as alterações necessárias para a conversão ser apresentada no formato `<nota-quantitativa> -> <nota-qualitativa>` no *standard output* (consola), quando a janela da aplicação for fechada.

Duração: 90 minutos
ISEL, 24 de Junho de 2022