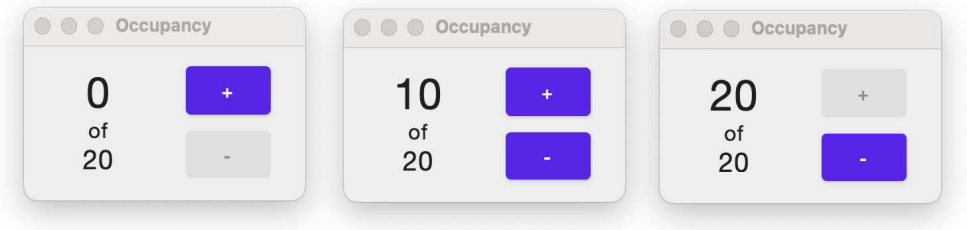
Grupo II

Pretende-se desenvolver uma aplicação para apoiar a regulação da ocupação de recintos fechados. Os recintos têm lotação máxima que, quando atingida, impede a entrada de pessoas até que outras saiam. Sempre que alguém entra ou sai do recinto, a lotação do momento é atualizada. A função `main` da aplicação é a que se apresenta de seguida.

```
fun main() = application {
  MaterialTheme {
    val state = WindowState(width= 250.dp, height= Dp.Unspecified)
    Window(onCloseRequest= ::exitApplication, state= state, title= "Occupancy") {
      MainContent()
    }
  }
}
```

A figura que se segue apresenta a janela da aplicação nos seus estados possíveis (da esquerda para a direita): o recinto está vazio; o recinto ainda não está lotado; o recinto está lotado.



1. [2] Defina o tipo `Occupancy`, cujas instâncias **imutáveis** representam a lotação de recintos. As instâncias deste tipo contêm a lotação atual (`current`) e a lotação máxima (`capacity`) do recinto. Na definição de `Occupancy` certifique-se que é impossível existirem instâncias com lotações máximas negativas ou com lotações atuais negativas ou superiores à lotação máxima (i.e. é lançada `IllegalArgumentException`). Este tipo deve ter também as propriedades `isFull` e `isEmpty` cujos valores são calculados a partir de `current` e `capacity`.

2. [2] Estenda o tipo `Occupancy` com as operações `increment` e `decrement`, que produzem a ocupação atualizada quando alguém entra ou sai do recinto. Caso alguma das operações viole as invariantes do tipo, é lançada a excepção `IllegalStateException`. Valorizam-se as soluções que não alterem a definição de `Occupancy` da alínea anterior.

3. [2] Crie os testes automáticos para verificação da correcção da definição de `Occupancy` e das operações `increment` e `decrement`. Para que a resposta não se torne demasiado exaustiva, implemente apenas um teste de utilização válida e outro de utilização inválida do tipo `Occupancy` e da operação `increment`.

4. [2] Implemente o *Composable* `OccupancyView`, sem estado interno (*stateless*), para apresentação da lotação actual e máxima (com aspecto aproximado à parte esquerda das janelas da figura).

5. [3] Implemente o *Composable* `MainContent`, com estado interno (*stateful*), que representa o conteúdo da janela principal da aplicação. Note que o estado dos botões (parâmetro `enabled` do *Composable* `Button`) reflete o estado de ocupação do recinto. Por simplificação, admita que a capacidade do recinto está *hard-coded* na implementação de `MainContent`.

6. [1] Descreva as alterações que realizaria à aplicação para tornar a capacidade do recinto parametrizável. Na descrição indique a forma de parametrização da execução que escolheu.

Duração: 90 minutos
ISEL, 24 de Janeiro de 2022