LUCAS HONDA TONINI

Lista 04 – Kotlin _Pontuando – Data de Entrega 24/04/2025

1)Receba um número entre 1 e 10. Calcule e mostre o resultado de uma tabuada.

```kotlin
fun tabuada(numero: Int) {
    println("Tabuada do $numero:")
    (1..10).forEach { i ->
        println("$numero x $i = ${numero * i}")
    }
}
fun main() {
    tabuada(22);
}
```

```
Tabuada do 22:
22 x 1 = 22
22 x 2 = 44
22 x 3 = 66
22 x 4 = 88
22 x 5 = 110
22 x 6 = 132
22 x 7 = 154
22 x 8 = 176
22 x 9 = 198
22 x 10 = 220
```

2) Nosso banco precisa tributar dinheiro de alguns bens que nossos clientes possuem. Para isso, vamos criar um sistema para isso.

a) Crie uma interface Tributavel que possui o método calculaTributos(), que retorna um **double**.

b) Alguns bens são tributáveis e outros não, **ContaPoupanca** não é tributável, já para **ContaCorrente** você precisa pagar 1% da conta e o **SeguroDeVida** tem uma taxa fixa de 42 reais.

c) As classes ContaCorrente e ContaPoupanca herdam de uma classe Conta. Essa classe Conta possui um saldo e os métodos sacar(double), depositar(double) e obterSaldo() que retorna o saldo da conta.

d) Vamos criar uma classe TestaTributavel com um método main() para testar o nosso exemplo.

```
class Conta(protected var saldo: Double) {
    fun sacar(valor: Double) {
        if (valor <= saldo) {
            saldo -= valor
            println("Saque de $valor realizado. Novo saldo: $saldo")
        } else {
            println("Saldo insuficiente")
        }
    }

    fun depositar(valor: Double) {
        saldo += valor
        println("Depósito de $valor realizado. Novo saldo: $saldo")
    }

    fun obterSaldo(): Double = saldo
}

class ContaCorrente(saldo: Double = 0.0) : Conta(saldo) {
    fun calculaTributos(): Double = obterSaldo() * 0.01
}
```

```
class ContaCorrente(saldo: Double = 0.0) : Conta(saldo) {
    fun calculaTributos(): Double = obterSaldo() * 0.01
}

class ContaPoupanca(saldo: Double = 0.0) : Conta(saldo) {
    fun calculaTributos(): Double = 0.0
}

class SeguroDeVida {
    fun calculaTributos(): Double = 42.0
}

fun main() {
    val contas = mutableListOf<Conta>()
    val seguros = mutableListOf<SeguroDeVida>()

    while (true) {
        println("\n=== Menu Principal ===")
        println("1. Criar Conta Corrente")
        println("2. Criar Conta Poupança")
        println("3. Adicionar Seguro de Vida")
        println("4. Realizar Operações Bancárias")
        println("5. Calcular Tributos")
```

```
        when (readlnOrNull()?.toIntOrNull()) {
            1 -> {
                print("Informe o saldo inicial: ")
                val saldo = readlnOrNull()?.toDoubleOrNull() ?: 0.0
                contas.add(ContaCorrente(saldo))
                println("Conta Corrente criada com sucesso!")
            }
            2 -> {
                print("Informe o saldo inicial: ")
                val saldo = readlnOrNull()?.toDoubleOrNull() ?: 0.0
                contas.add(ContaPoupanca(saldo))
                println("Conta Poupança criada com sucesso!")
            }
            3 -> {
                seguros.add(SeguroDeVida())
                println("Seguro de Vida adicionado!")
            }
            4 -> {
                if (contas.isEmpty()) {
                    println("Nenhuma conta cadastrada!")
                    continue
                }
```

```
                println("\n=== Operações Bancárias ===")
                contas.forEachIndexed { index, conta ->
                    println("${index + 1}. ${conta::class.simpleName} - Saldo: ${conta.obterSaldo()}")
                }

                print("Selecione uma conta: ")
                val contaIndex = readlnOrNull()?.toIntOrNull()?.minus(1) ?: continue

                if (contaIndex !in contas.indices) {
                    println("Índice inválido!")
                    continue
                }

                val conta = contas[contaIndex]

                println("\n1. Sacar")
                println("2. Depositar")
                println("3. Ver Saldo")
                print("Escolha a operação: ")

                when (readlnOrNull()?.toIntOrNull()) {
                    1 -> {
                        print("Valor para sacar: ")
```

```
                        print("Valor para sacar: ")
                        val valor = readlnOrNull()?.toDoubleOrNull() ?: continue
                        conta.sacar(valor)
                    }
                    2 -> {
                        print("Valor para depositar: ")
                        val valor = readlnOrNull()?.toDoubleOrNull() ?: continue
                        conta.depositar(valor)
                    }
                    3 -> println("Saldo atual: ${conta.obterSaldo()}")
                    else -> println("Opção inválida!")
                }
            }
            5 -> {
                println("\n=== Tributos Calculados ===")

                // Tributos de contas
                contas.forEach { conta ->
                    when (conta) {
                        is ContaCorrente -> println("Conta Corrente: ${conta.calculaTributos()}")
                        is ContaPoupanca -> println("Conta Poupança: ${conta.calculaTributos()}")
                    }
                }
```

3) Conforme o diagrama de classe abaixo desenv