ESCOLA
SUPERIOR
DE MEDIA
ARTES
E DESIGN
POLITÉCNICO
DO PORTO

ALP

PROVA DE AVALIAÇÃO

ATIVIDADE LETIVA

Tecnologias e Sistemas de Informação para a Web

CURSO

2022/2023	2023/06/07	14h00 / 16h00	1h45
ANO LETIVO	DATA	HORA	DURAÇÃO

Programação Orientada a Objetos	1º ano
UNIDADE CURRICULAR	ANO

Mário Pinto / Ricardo Queirós/Miguel Correira/Daniel Carneiro	Avaliação distribuída
DOCENTE	ÉPOCA

Observações:

- O teste é individual e deve ser resolvido através da linguagem JavaScript.
- Podem consultar **apenas** as seguintes referências externas: MDN, W3Schools, Moodle e GitHub (repo da UC e repo pessoal)
- A avaliação do teste poderá implicar convocar os estudantes para uma sessão **de avaliação oral**, onde podem ter de explicar o código entregue
- Serão usadas ferramentas anti-plágio na avaliação dos testes
- Descarregue a pasta **NR_ALUNO** do Moodle. Altere o nome da pasta para o **seu número de aluno**.
- **No final do teste submeta a pasta (compactada) no moodle, em objeto próprio de submissão**

Crie um programa de gestão de contas bancárias. O programa é constituído por 4 ficheiros já criados:

- **index.html** (inclui toda a interface gráfica)
- **index.js** (gere os eventos do rato e invoca os métodos da classe **GestorContas**)
- **contaBancaria.js** (informação sobre uma conta bancária individual)
- **gestorContas.js** (informação sobre as contas bancárias existentes e funções sobre as contas)

Requisitos do programa:

1. Na classe **ContaBancaria**
    a. Inclua as seguintes propriedades a serem inicializadas no construtor da classe: Número da conta (privado), Nome do titular da conta (privado), Saldo da conta (privado e com o valor 0 por omissão)

    b. Crie os métodos "get" e "set" para as propriedades da classe **ContaBancaria**. Note que para algumas propriedades poderá só ter lógica o método “get”

    c. Crie os métodos
        i. **depositar(valor)** - que recebe um valor e incrementa o saldo
        ii. **levantar(valor)** - que recebe um valor e que verifica se o saldo é suficiente. Se não for suficiente, informa o utilizador. Caso contrário, debita o valor ao saldo atual e informa igualmente o utilizador.

2. Na classe **GestorContas**:
    a. Inclua a propriedade a ser inicializada no construtor da classe (sem parâmetros):
        i. Lista de contas (array vazio de objetos **ContaBancaria**)

    b. Crie os métodos:
        i. **criarConta()**: que criar uma nova conta bancária e adicionar ao array de contas (certifique-se de que o número da conta seja único, caso contrário deve gerar uma mensagem de conta já existente). O método deve receber, recorde-se, o número da conta, o nome do titular e o saldo inicial, que por omissão é 0.

        ii. **listarContas()**: que devolve lista de todas as contas existentes no array (número da conta e nome do titular). As contas devem ser listadas numa tabela criada dinamicamente (preferencialmente no index.js).

        iii. **realizarDeposito()**: que realiza um depósito de um valor numa conta específica. O método deve receber como argumentos o número da conta e valor. Deve validar se a conta existe, e se sim, depositar esse valor na conta.

        iv. **realizarLevantamento()**: que realiza um levantamento de um valor de uma conta específica. Validar se conta existe e se saldo é suficiente.

        v. **exibirSaldo()**: que exibe o saldo de uma conta específica, passada como argumento do método. Verificar se conta existe.

3. No ficheiro **index.js** inclua todo o código necessário para gerir os eventos do rato (cliques nos botões) e respetivas chamadas aos métodos codificados na classe **GestorConta**.

Notas importantes:

- Deve conectar todos os ficheiros entre si usando as funcionalidades de **import/export**
- Para todos os métodos criados exiba numa **caixa de alerta** os resultados e as notificações de sucesso e insucesso