1. Classe `Pessoa` e Herança

- **Problema**: Crie uma classe `Pessoa` com os seguintes atributos:
  - `nome` (string),
  - `idade` (número),
  - `sexo` (string).
    Adicione métodos:
  - `apresentar()`: retorna uma string com as informações formatadas, como "Olá, meu nome é João, tenho 25 anos e sou homem".
- Crie uma classe `Aluno` que herde de `Pessoa` e adicione:
  - `matricula` (número),
  - `curso` (string).
    O método `apresentar()` deve incluir informações sobre o curso.
- **Extra**: Crie uma instância de `Aluno` e invoque o método `apresentar()`.

---

2. Classe `ContaBancaria`

- **Problema**: Crie uma classe `ContaBancaria` com os seguintes atributos:
  - `titular` (string),
  - `saldo` (número).
    Adicione os métodos:
  - `depositar(valor)`: aumenta o saldo.
  - `sacar(valor)`: diminui o saldo se houver saldo suficiente; caso contrário, exiba uma mensagem de erro.
  - `mostrarSaldo()`: exibe o saldo atual.
- **Extra**:
  - Crie uma classe `ContaCorrente` que herde de `ContaBancaria` e adicione um atributo `limite` (número).
  - Modifique o método `sacar()` para permitir saque até o valor do limite.

---

3. Sistema de Gerenciamento de Produtos

- **Problema**: Crie uma classe `Produto` com:
  - `nome` (string),
  - `preco` (número),
  - `quantidadeEmEstoque` (número).
    Adicione os métodos:
  - `atualizarEstoque(quantidade)`: aumenta ou reduz o estoque.
  - `calcularValorEstoque()`: retorna o valor total do estoque (preço × quantidade).
- **Extra**:
  - Crie uma subclasse `ProdutoPerecivel` que inclua um atributo `dataDeValidade` (string) e um método `verificarValidade(dataAtual)` para retornar se o produto ainda está válido.

---

4. Classe `Veiculo`

- **Problema**: Crie uma classe `Veiculo` com os atributos:
  - `marca` (string),
  - `modelo` (string),
  - `ano` (número).
    Adicione métodos:
  - `descrever()`: retorna uma string descrevendo o veículo, como "Marca: Ford, Modelo: Fiesta, Ano: 2020".
- Crie uma classe `Carro` que herde de `Veiculo` e adicione:
  - `portas` (número).
    O método `descrever()` deve incluir o número de portas.
- **Extra**: Crie uma subclasse `Moto` que herde de `Veiculo` e inclua:
  - `cilindradas` (número).
    Modifique o método `descrever()` para incluir esse atributo.

---

5. Classe de Gerenciamento de Funcionários

- **Problema**: Crie uma classe `Funcionario` com:
  - `nome` (string),
  - `salario` (número).
    Adicione métodos:
  - `aumentarSalario(percentual)`: aumenta o salário com base em um percentual.
  - `mostrarInformacoes()`: exibe o nome e salário do funcionário.
- **Extra**:
  - Crie uma subclasse `Gerente` que adicione:
    - `departamento` (string).
      Modifique o método `mostrarInformacoes()` para incluir o departamento.
  - Crie outra subclasse `Estagiario` que limite o aumento salarial a no máximo 10%.

---

6. Sistema de Biblioteca

- **Problema**: Crie uma classe `Livro` com os atributos:
  - `titulo` (string),
  - `autor` (string),
  - `disponivel` (boolean).
    Adicione métodos:
  - `emprestar()`: altera `disponivel` para `false`.
  - `devolver()`: altera `disponivel` para `true`.
  - `status()`: exibe o título e se está disponível ou emprestado.

- **Extra**:
  - Crie uma classe `Biblioteca` que gerencie uma coleção de livros.
  - Adicione métodos para listar os livros disponíveis e buscar um livro pelo título.

---

7. Classe `Jogador` para Jogos Online

- **Problema**: Crie uma classe `Jogador` com:
  - `nome` (string),
  - `nivel` (número),
  - `experiencia` (número).
    Adicione os métodos:
  - `ganharExperiencia(pontos)`: aumenta a experiência.
  - `subirDeNivel()`: aumenta o nível se a experiência atingir 100.
- **Extra**:
  - Crie uma subclasse `Guerreiro` que adicione:
    - `forca` (número).
      Aumente a `forca` quando o jogador subir de nível.

---

8. Classe `Turma` e Alunos

- **Problema**: Crie uma classe `Turma` com os atributos:
  - `curso` (string),
  - `alunos` (array de strings).
    Adicione métodos:
  - `adicionarAluno(nome)`: adiciona um aluno à turma.
  - `removerAluno(nome)`: remove um aluno pelo nome.
  - `listarAlunos()`: lista todos os alunos da turma.
- **Extra**:
  - Adicione uma subclasse `TurmaOnline` com o atributo adicional `linkDeAcesso`.
  - Modifique o método `listarAlunos()` para incluir o link de acesso.

---

9. Classe para Gerenciar Tarefas

- **Problema**: Crie uma classe `Tarefa` com os atributos:
  - `descricao` (string),
  - `concluida` (boolean).
    Adicione os métodos:
  - `marcarConcluida()`: altera `concluida` para `true`.
  - `descrever()`: exibe a descrição e o status da tarefa.
- **Extra**:

    - Crie uma classe `ListaDeTarefas` para gerenciar um conjunto de objetos `Tarefa`.
    - Adicione métodos para:
        - Adicionar uma tarefa.
        - Listar todas as tarefas concluídas.

---

10. Classe para Controle de Estacionamento

- **Problema**: Crie uma classe `Carro` com os atributos:
    - `placa` (string),
    - `modelo` (string).
      Adicione métodos:
    - `descrever()`: exibe a placa e o modelo.
- Crie uma classe `Estacionamento` com:
    - `vagasTotais` (número),
    - `carros` (array de objetos `Carro`).
      Adicione métodos:
    - `adicionarCarro(carro)`: adiciona um carro, se houver vaga.
    - `removerCarro(placa)`: remove um carro pela placa.
    - `listarCarros()`: exibe todos os carros estacionados.

Esses exercícios ajudam a explorar conceitos fundamentais de **classes, herança, métodos e encapsulamento** no ES6, enquanto trabalham cenários práticos do dia a dia.