**Lista de exercícios**

**Tipos abstrato de dados**

1. O que é um tipo concreto de dado?

**R:** Um tipo concreto de dado é uma estrutura de dados cuja implementação interna é conhecida e acessível, permitindo que o usuário a manipule diretamente. Ele define não apenas as operações, mas também a forma como os dados são organizados e armazenados na memória.

2. O que é um tipo abstrato de dados?

**R:** Um tipo abstrato de dados (TAD) é um modelo matemático para tipos de dados que se concentra no comportamento e nas operações que podem ser realizadas com os dados, em vez de sua implementação específica.

3. O tipo int do Python é um TAD? Explique.

**R:** Sim, o tipo “int” em Python pode ser considerado um TAD, porque ele define operações que podem ser executadas pelos programadores sem expor a sua implementação interna.

4. Qual é a diferença na forma que o TAD é manipulado por quem implementa e por quem usa o TAD?

**R:** Para quem implementa o TAD, é possível acessar e manipular diretamente a estrutura de dados interna e controlar como os dados são armazenados e processados. Ou seja, o implementador define a lógica interna do TAD e como as operações serão realizadas. Por outro lado, quem utiliza o TAD só tem acesso à interface pública fornecida pelo TAD (funções e métodos disponíveis), sem poder interagir diretamente com a implementação interna.

5. Quais são as vantagens e desvantagens dos TAD 's?

**R:** Vantagens:

- Facilita o reuso criando abstrações;
- Facilita a verificação permitindo que cada TAD seja testado de forma isolada;
- Aumenta a confiabilidade mantendo os valores em estado consistente;
- etc

Desvantagens:

- A especialização requer um investimento inicial maior;
- Pode aumentar a complexidade adicionando abstrações desnecessárias;
- Pode gerar perda de desempenho devido às construções de abstrações;

6. Como definimos TAD 's em Python?

**R:** Em Python, definimos TADs usando a construção de classes através da palavra-chave **class.** Ao criar uma classe, podemos definir atributos (dados) e métodos (operações) que encapsulam o comportamento e as operações permitidas sobre esses dados.

7. O que é um construtor?

**R:** Um construtor é um método especial em uma classe que é executado automaticamente quando um objeto da classe é criado. Ele é responsável por inicializar as variáveis de instância (atributos) da classe, atribuindo valores iniciais e configurando o estado inicial do objeto de acordo com a implementação do programador.

8. Considere o TAD Contador especificado no arquivo contador_tad.py e:
    a. Faça uma cópia do arquivo contador_tad.py chamada contador_int.py e implemente o TAD usando um inteiro para armazenar o valor do contador.

       **R:** Feito no VsCode e postado no GitHub

b. Faça uma cópia do arquivo contador_tad.py chamada contador_alt.py e implemente o TAD usando uma representação interna diferente da do exercício anterior, não use operações aritméticas na implementação.
**R:** Feito no VsCode e postado no GitHub

c. Compare as duas implementações e discuta qual é a mais adequada.

**R:** A **primeira implementação** é mais adequada porque utiliza operações naturais e diretas sobre inteiros, tornando o código mais simples, legível e eficiente. A segunda implementação, embora interessante como exercício de abstração, não é ideal para resolver o problema proposto de maneira eficiente.

9. Considere o TAD Dias especificado no arquivo dias_tad.py e:
   a. Faça uma cópia do arquivo dias_tad.py chamada dias_bools.py e implemente o TAD usando 7 campos booleanos, cada um indicado se um dia está no conjunto de dias ou não.
   **R:** Feito no VsCode e postado no GitHub

   b. Faça uma cópia do arquivo dias_tad.py chamada dias_alt.py e implemente o TAD usando uma representação interna diferente da do exercício anterior.
   **R:** Feito no VsCode e postado no GitHub

   c. Compare as duas implementações e discuta qual é a mais adequada.

**R:** A **segunda implementação** é a mais adequada devido a sua legibilidade, clareza, eficiência e manutenção.

10. Considere o TAD Selecao especificado no arquivo selecao_tad.py e:
    a. Faça uma cópia do arquivo selecao_tad.py chamada selecao_inicio_fim.py e implemente o TAD usando três campos: a célula inicial, o início e o fim do intervalo.
    **R:** Feito no VsCode e postado no GitHub

b. Faça uma cópia do arquivo selecao_tad.py chamada selecao_alt.py e implemente o TAD usando uma representação interna diferente da do exercício anterior.
**R:** Feito no VsCode e postado no GitHub

c. Compare as duas implementações e discuta qual é a mais adequada.

**R:** A **primeira implementação** é mais adequada, pois a legibilidade é melhor, o código é mais fácil de entender e manter, e há menos risco de erro ao manipular as colunas da seleção. Embora a **segunda implementação** possa parecer mais compacta, ela pode introduzir complexidade desnecessária, especialmente se o código crescer ou se for necessário trabalhar com outros desenvolvedores.

11. Seu amigo disse que não é possível implementar a classe Robo de forma diferente da que vimos em sala. Mostre que ele está errado fazendo uma implementação que usa apenas um campo do tipo string que armazena a info (nome seguido da posição entre parênteses) do robô. Crie uma cópia do arquivo robo_tad.py chamada robo_info.py e escreva a implementação nesse novo arquivo. Compare essa implementação com a feita em sala e discuta qual é a mais adequada.

**R:** Feito no VsCode e postado no GitHub.

A **primeira implementação** é a mais adequada, pois separa o nome e a posição do robô em variáveis distintas, o que melhora a organização, facilita a leitura e torna o código mais fácil de manter. Com os dados armazenados separadamente, é mais simples realizar modificações e manipulações, além de reduzir a chance de erros. Já na **segunda implementação**, onde o nome e a posição estão combinados em uma única string, a manipulação dos dados se torna mais complexa, exigindo parsing e conversões constantes. Isso dificulta a manutenção e aumenta a probabilidade de problemas, especialmente em projetos maiores.

12. Em algumas empresas é comum o uso de um banco de horas, que é um esquema de compensação de horas extras. Quando um funcionário trabalha além da sua jornada normal, as horas e minutos extras trabalhados são “depositados” em um banco de horas. Posteriormente o funcionário pode usar o saldo no banco de horas para se ausentar do trabalho.

    a. Projete um TAD BancoHoras com operações para depósito de horas e minutos, “saque” de horas e minutos (o saldo deve ser suficiente) e consulta de saldo (que deve gerar uma string no formato HH:MM). Em qualquer operação a quantidade de minutos não podem ser maior que 60
    **R:** Feito no VsCode e postado no GitHub

    b. Faça uma implementação do TAD usando um campo para armazenar o saldo em minutos.
    **R:** Feito no VsCode e postado no GitHub

    c. Faça uma implementação do TAD usando dois campos, uma para horas e outro para minutos.
    **R:** Feito no VsCode e postado no GitHub

    d. Compare as duas implementações e discuta qual é a mais adequada.

**R:** A **primeira implementação** (com o saldo total em minutos) é mais adequada, especialmente se o foco for em simplicidade e facilidade de manutenção. A separação de horas e minutos (como na **segunda implementação**) pode parecer mais intuitiva à primeira vista, mas acaba tornando o código mais propenso a erros e mais difícil de escalar, especialmente quando se lida com cálculos envolvendo tempo.