Centro Universitário Senac Santo Amaro

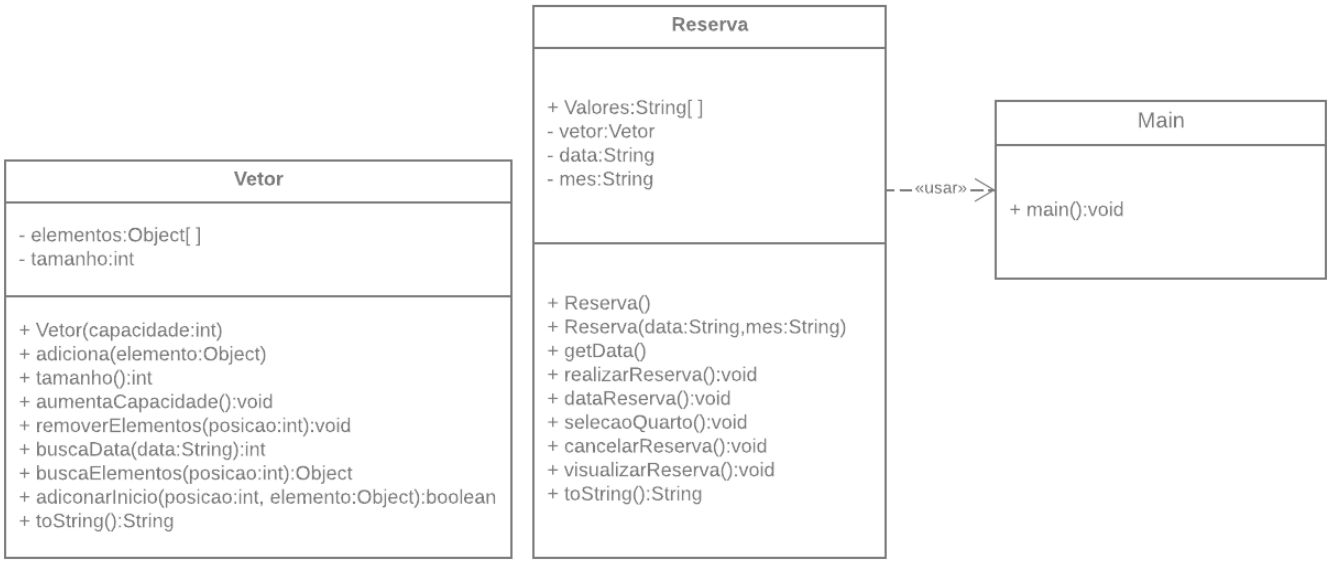
Implementação de Encapsulamento

Nome: Eric Paixão Andrade e Leonardo Gama Pereira

Professor: Carlos Verissimo

São Paulo

2023



Encapsulamento da classe Reserva:

Atributos Privados: Os atributos ‘data’ e ‘mês’ são declarados como privados com os modificadores de acesso ‘private’. Isso significa que esses atributos não podem ser acessados diretamente de fora da classe ‘Reserva’. Isso é uma prática de encapsulamento, pois oculta o estado interno da classe e restringe o acesso direto aos atributos.

```
private String data;
private String mes;
```

Métodos Públicos de Acesso: A classe ‘Reserva’ fornece métodos públicos para acessar e manipular os atributos privados ‘data’ e ‘mes’. Os métodos ‘getData()’ permitem obter o valor desses atributos, enquanto os construtores e outros métodos, como ‘realizarReserva()’, ‘dataReserva()’, ‘selecaoQuarto()’, ‘cancelarReserva()’, ‘visualizarReservas()’, fornecem funcionalidades relacionadas ao objeto Reserva.

```
public String getData() {
    return data;
}
```

Ocultação de Detalhes Internos: Os detalhes internos de como uma reserva é manipulada, como a lógica para realizar uma reserva, cancelar uma reserva ou visualizar reservas, são encapsulados dentro da classe Reserva. Os métodos públicos são a única maneira de interagir com esses detalhes internos, fornecendo uma interface controlada para o mundo externo

Utilização do Atributo Privado vetor: O atributo privado vetor, que é uma instância da classe Vetor, também segue o encapsulamento, pois não é diretamente acessível fora da classe Reserva. A classe Reserva pode interagir com vetor por meio de métodos da classe Vetor, presumivelmente encapsulados dentro dela.

```
private static Vetor vetor = new Vetor(10);
```

Em resumo, o código demonstra boas práticas de encapsulamento ao ocultar detalhes internos da classe Reserva, fornece métodos públicos para acesso controlado aos atributos e comportamentos e tratar exceções para manter a robustez do sistema sem expor detalhes de implementação.

Encapsulamento da classe Vetor:

Atributos Privados: Os atributos ‘elementos’ e ‘tamanho’ são declarados como privados com o modificador de acesso ‘private’. Isso significa que eles não podem ser acessados diretamente fora da classe ‘Vetor’, contribuindo para o encapsulamento, pois oculta o estado interno da classe.

```
private Object[] elementos;
private int tamanho;
```

Métodos Públicos de Acesso e Comportamento: A classe Vetor fornece métodos públicos para acessar e manipular seus atributos e comportamentos. Isso inclui métodos para adicionar elementos, verificar o tamanho, aumentar a capacidade, remover elementos, buscar elementos por posição e buscar elementos por data. Todos esses métodos são a interface controlada para interagir com a classe.

Tratamento de Exceções: Quando ocorrem erros, como tentar adicionar um elemento em um vetor cheio ou acessar uma posição inválida, a classe Vetor lança exceções e as trata internamente. Isso ajuda a manter a robustez do código e oculta os detalhes de implementação de como essas exceções são tratadas.

Método toString(): A classe Vetor substitui o método toString() para fornecer uma representação em string de seus elementos. Isso é uma prática comum para facilitar a visualização e depuração dos objetos.

Operações de Manipulação de Vetor: A classe Vetor encapsula várias operações comuns de manipulação de vetores, como adicionar, remover e buscar elementos. Essas operações são executadas internamente na classe, e os detalhes de implementação são ocultos do código cliente.

O encapsulamento aqui é evidenciado pelo fato de que os detalhes internos da estrutura de dados do vetor (como o array elementos) e sua manipulação são abstraídos e controlados por meio de métodos públicos. Isso permite que o código Reserva use a classe Vetor de forma segura, sem a necessidade de conhecer ou manipular diretamente os atributos internos do vetor.