Aplicação de Paradigma de Orientação a Objetos: Sistema FMC com MCDU

1. Objetivo e Cenário de Aplicação:

Objetivo Estabelecido: Integrar a paixão pela aviação com a ciência da computação, desenvolvendo um Flight Management Computer (FMC).

Cenário de Aplicação: O sistema será um FMC integrado com um Flight Management System (FMS) que culmina em uma Multi-Function Control Display Unit (MCDU) utilizada para simular operações de voo.

2. Diferenciar entre Modelar e Implementar:

Modelagem: Criar uma representação conceitual do FMC, FMS e MCDU e de como eles interagem no contexto da aviação.

Implementação: Desenvolver um software que simula as funções do FMC e permite interações através do MCDU.

3. Identificação de Classes:

Com base nos arquivos fornecidos, podemos adicionar mais classes:

Classe **NavObject**: Representa um objeto de navegação genérico no sistema.

Classe **Database**: Gerencia a base de dados de navegação.

Classe **XPlaneReceiver**: Interage com o simulador de voo X-Plane.

Classe **Avionics**: Representa os aviónicos da aeronave, incluindo informações como posição, velocidade e altitude.

Classe **ACARS**: Gerencia a funcionalidade do sistema ACARS.

Classe **ATC**: Gerencia a interação com o controle de tráfego aéreo.

4. Defina Atributos e Métodos:

Para algumas das novas classes:

NavObject: Atributos como **type**, **latitude**, **longitude**. Métodos como **calculateDistanceTo()**, **calculateCourseTo()**.

Database: Atributos como **airports**, **navaids**, **waypoints**. Métodos como **loadDatabase()**, **findClosestAirport()**.

XPlaneReceiver: Atributos como **datarefs**, **socket**. Métodos como **request()**, **run()**.

Avionics: Atributos como **position**, **speed**, **altitude**. Métodos como **update()**, **analyze()**.

ACARS: Atributos como **messages**, **status**. Métodos como **run()**, **fetch_messages()**, **parse_message()**.

ATC: Atributos como **api**, **midn**, **act**, **next**. Métodos como **logon()**, **logoff()**, **send_message()**.

5. Estabelecer Relações entre Classes:

Algumas das relações podem ser:

FMC utiliza **Database** para obter informações de navegação.

FMC se comunica com **XPlaneReceiver** para interagir com o simulador de voo.

FMC utiliza **Avionics** para obter informações sobre o estado da aeronave.

FMC pode enviar e receber mensagens via **ACARS** e **ATC**.

7. Implementação e Código:

A implementação foi feita em Python e utiliza várias classes para representar diferentes componentes do sistema. O código é modular e orientado a objetos, com cada classe tendo responsabilidades bem definidas.

Bônus:

O projeto foi integrado ao simulador de voo X-Plane, podendo assim demonstrar a comunicação do algoritmo com uma aplicação prática.

Nota: Este projeto não é apenas uma aplicação prática de conceitos da área de ciência da computação, mas também uma realização pessoal ao combinar minha paixão pela aviação com minha área de estudo.

Aluno: *Henrique Martins Lacerda*

RA: *22203112*