Trabalho de Programação

Pablo Freitas Santos
Engenharia Mecatrônica
Universidade Federal de Santa Catarina
Joinville, Brasil
pablo.freitas@grad.ufsc.br

***Resumo*—Esse trabalho tem objetivo de implementar um algoritmo de uma fila em que representa a pista de pouso e decolagem de um aeroporto e trabalhando o conceito de classes e construir um diagrama do algoritmo.**

***Index Terms*—classe, fila, diagrama**

I. Introdução

Para a construção da classe `Extended_queue` inicialmente pensou-se na organização de memória separada para cada função e em seguida como liberar para evitar vazamentos. Depois um diagrama de classes de acordo com o UML e com o objetivo de separar a acessibilidade e os tipos de relacionamentos que cada classe possui.

II. Desenvolvimento

A partir da inclusão das bibliotecas foi feito o construtor da `Extended_queue`, que basicamente inicializava a fila da forma padrão, primeiro e último nó nulos e tamanho igual a zero. Em seguida no destrutor foi chamada a função `Extended_queue::serve()` enquanto a fila não estivesse vazia, evitando-se assim vazamentos de memória.

A maiorias das funções não houve problemas de implementação, no entanto `Error_code Extended_queue::append(Plane &plane)` teve a necessidade de pensar nos modos que poderiam corromper o algoritmo para se adequar aos retornos e `Error_code` e avaliar os casos de colocar em uma fila vazia ou colocar em uma fila que já possui elementos.

Na construção do diagrama(Figure 1) foi utilizado a ferramenta Umbrello [2], e no caso desse algoritmo só foi encontrado 2 tipos de associações entre classes, agregação simples(losango vazado), demonstra que as informações de um objeto precisam ser complementadas por um objeto de outra classe (objeto - todo) e composição (losango preenchido), que representa um vínculo mais forte entre objetos-todo e objetos-parte. Objetos-parte têm que pertencer ao objeto-todo [3].

III. Conclusão

Nesse projeto pode ser observado que cada classe possui uma importância para o algoritmo e devemos levar em conta a relação com as outras classes para construir diagramas que representando a acessibilidade de cada classe e separar cada classe por função.

Figura 1. Diagrama de classes do aeroporto

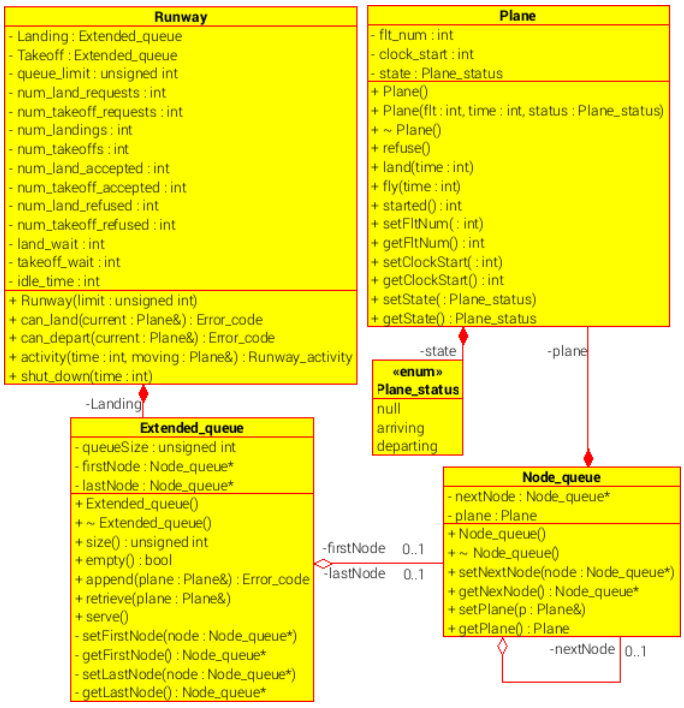
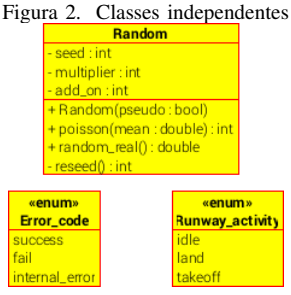


Figura 2. Classes independentes



Referências

[1] Paul Deitel and Harvey Deitel. 2013. C++ how to Program (Early Objects Version) Deitel (9th ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.
[2] https://umbrello.kde.org/
[3] http://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/uml-diagrama-classes-relacionamentos_v01.pdf