Unidade 2 - Introdução à GUI: Arquitetura de Software

1       ESTRUTURA DE UMA APLICAÇÃO OO

A **boa prática de programação** recomenda que o programador deva **separar** a PARTE LÓGICA do programa da INTERFACE COM O USUÁRIO.

Assim, em nossas aplicações orientados a objetos SEMPRE teremos no mínimo **2 classes**:

-      **CLASSE PRINCIPAL**: responsável pela interface com o usuário (leitura e escrita dos dados) e criação dos objetos necessários para a solução do problema. Doravante esta classe será denominada de **main**;

-      **CLASSE DO MODELO**: responsável por executar as tarefas que solucionam a situação-problema proposta.

Neste caso dizemos que há uma relação entre as classes.

Deste modo, usaremos o seguinte modelo de arquitetura em nossas aplicações:

 A CLASSE PRINCIPAL terá sempre as seguintes tarefas fundamentais:

          i.              CRIAR O(S) OBJETO(S) da(s) classe(s) do modelo necessário(s) para a solução do problema;

        ii.              OBTER do usuário, através dos dispositivos de entrada, OS DADOS necessários para inicializar a classe do modelo;

       iii.              PASSAR OS DADOS obtidos para a classe do modelo;

       iv.              CHAMAR OS MÉTODOS da classe do modelo necessários para a solução do problema proposto;

        v.              OBTER da classe do modelo OS RESULTADOS do problema;

       vi.              INFORMAR ao usuário através dos dispositivos de saída OS RESULTADOS obtidos;

A CLASSE DO MODELO terá sempre as seguintes tarefas fundamentais:

      vii.              RECEBER os dados da classe principal;

     viii.              PROCESSAR os dados recebidos;

       ix.              DISPONIBILIZAR os resultados do processamento para a classe principal.

 Esta é uma simplificação da ideia de desenvolver um software em camadas. Uma camada é a maneira de macro-organização de um software, seguindo uma arquitetura de software.

2       ARQUITETURA DE SOFTWARE

Sem entrar em discussões conceituais de Arquitetura de Software (que será alvo em disciplinas posteriores do curso), vamos entender que a construção de um software também passa por planejar sua estrutura, que é um dos aspectos da sua arquitetura.

Atualmente uma das abordagens empregadas para estruturar um software consiste em dividi-lo em diferentes camadas, cada qual fornecendo serviços para as camadas adjacentes, procurando deixar uma camada mais independente possível da outra camada (baixo acoplamento).

Como você pode observar na figura a seguir, um software pode ser divido em basicamente três camadas: apresentação, domínio da aplicação e acesso a dados.

* A **camada de domínio da aplicação**, também conhecida como camada de negócio ou de objetos de negócio ou de modelo, é aquela onde estão localizadas as classes que fazem parte do domínio do problema, ou seja, classes correspondentes a objetos que fazem parte da descrição do problema. Estas classes não são inventadas ou criadas - pelo contrário, são **identificadas** na descrição do problema, pois já existem naturalmente no entendimento a respeito do problema. Estas classes existem independentemente da tecnologia que vai ser empregada para armazenar os dados ou para implementar a interface com o usuário.
* A **camada de apresentação** se preocupa em oferecer uma interface ao usuário. Assim, em determinado software sua camada de apresentação pode ser implementada para a interface caracter. Também pode ter outra versão do software implementando a apresentação para web, ou para interface gráfica, ou para *mobile* e assim por diante. É importante observar que a camada de apresentação pode ser modificada sem que se modifique o que o usuário está esperando que o software faça. Portanto, sem modificar a camada de domínio da aplicação.
* A **camada de acesso aos dados**vai fornecer serviços para que os dados possam ser armazenados e recuperados em algum meio específico. Esta camada vai ser dependente da opção utilizada para este armazenamento: arquivos, bancos de dados, bancos de objetos, etc. Porém, a maneira de armazenar os dados também não afeta o que o usuário está esperando que o software faça. Portanto, sem modificar a camada de domínio da aplicação.

Cada camada vai ser planejada e construída procurando oferecer os serviços para a camada adjacente, sendo isoladas/independentes. Mas como se garante independência, minimizando o acoplamento entre elas?

Fazendo com que cada camada cuide de suas atividades e não se intrometa nas atividades das outras. Por exemplo, a camada de negócios deve realizar o processamento das operações sem que haja qualquer interação com usuário - portanto, não haverá nenhum comando ou instrução de entrada/saída, pois isto é responsabilidade da camada de apresentação. Da mesma maneira, a camada de negócios não terá comandos ou instruções de acesso a dados - responsabilidade das classes que estão na camada de dados. Por sua vez, a camada de apresentação deve evitar ter código para realizar cálculos e processamento - ela deve solicitar o processamento para a camada de negócio, fornecendo as entradas do usuário e exibindo os resultados.

Esta divisão básica em três camadas é apenas o início de uma arquitetura. Cada software poderá ter ainda mais camadas dependendo de sua complexidade e de seus requisitos.

Porém o desenvolvimento de uma solução para o problema inicia normalmente na camada de domínio da aplicação.

E como se identificam as classes que compõem esta camada ? Isto será objeto de estudo logo, logo.