Encapsulamento, Herança e Composição e Polimorfismo

1. O item 00 está descrito no moodle

2. O mundo das formas é muito rico. Cria uma interface (em C++ uma classe abstract apenas com métodos virtuais = 0) chamada de `Forma`. A inteface mais abstrata, `Forma`, deve ter dois métodos: `std::string get_cor()` e `std::string get_nome()`. A mesma tem duas sub-interfaces: `Forma2D` e `Forma3D`. A primeira, deve conter os métodos `double get_area() e double get_perimetro()`. A segunda tem um único método `double get_volume()`. Implemente as classes: `Quadrado, Circulo, Triangulo, Esfera e Cubo`.

3. Os serviços de correio expresso, como FedEx, DHLe UPS, oferecem várias opções de entrega, cada qual com custos específicos. Crie uma hierarquia de herança para representar vários tipos de pacotes. Utilize `Package` como a classe básica da hierarquia, então inclua as classes `TwoDayPackage` e `OvernightPackage` que derivam de `Package`. A classe básica `Package` deve incluir membros de dados que representam nome e endereço. Para simplificar nosso código, represente o endereço como uma única string. Além dos membros de citados anteriormente, armazene dados que armazenam o peso (em quilos) e o custo por quilo para a entrega do pacote. O construtor `Package` deve inicializar todos os membros de dados, em outras palavras, todos são argumentos do construtor. Assegure que o peso e o custo por quilo contenham valores positivos (faça uso de unsigned int). `Package` deve fornecer um método `public calculate_cost` que retorna um `double` indicando o custo associado com a entrega do pacote. A função `calculate_cost` de `Package` deve determinar o custo multiplicando o peso pelo custo (em quilos). A classe derivada `TwoDayPackage` deve herdar a funcionalidade da classe básica `Package`, mas também incluir um membro de dados que representa uma taxa fixa que a empresa de entrega cobra pelo serviço de entrega de dois dias. O construtor `TwoDayPackage` deve receber um valor para inicializar esse membro de dados. `TwoDayPackage` deve redefinir o método `calculate_cost` para que ela calcule o custo de entrega adicionando a taxa fixa ao custo baseado em peso calculado pela função `calculate_cost` da super classe Package. A classe `OvernightPackage` deve herdar diretamente da classe `Package` e conter um membro de dados adicional para representar uma taxa adicional por quilo cobrado pelo serviço de entrega noturno. `OvernightPackage` deve redefinir o método `calculate_cost` para que ela acrescente a taxa adicional por quilo ao custo-padrão por quilo antes de calcular o custo da entrega.

4. Use a hierarquia de herança `Package` criada no exercício anterior para criar uma classe `DestinationPackages` armazena para cada destinatário os seus `Package`s. A classe deve conter um map de vectors (`std::map<string, std::vector<Package*>>`) que armazena um vetor do tipo mais abstrato `Package` no valor, indexando pelo nome do destinatário na chave. Note que o vetor é de ponteiros para fazer uso de polimorfismo. Sua classe deve ter um método `void add_package(Package g)` que guarda um novo pacote no mapa. A mesma, também deve ter dois métodos com o mesmo nome `double custo_total()` que realiza um loop pelo para processar o `Package`s polimorficamente. A primeira variação `double custo_total()` computa o custo de todos os usuários, enquanto a segunda `double custo_total(string)` computa os custos de um único usuário.

Referências

[1] D.J. Barnes and M. Kölling. *Programação orientada a objetos com java: uma introdução prática usando Blue J.* PRENTICE HALL BRASIL, 2004.

[2] P.J. Deitel and H.M. Deitel. *C++ how to Program.* Deitel series. Pearson Prentice Hall, 2010.

[3] G.L. McDowell. *Cracking the Coding Interview: 189 Programming Questions and Solutions.* CareerCup, LLC, 2015.