Data: 04/10/2010

Prof. Dr. Jean Marcelo Simão

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ - Campus de Curitiba (Brasil) - **Departamento Acadêmico de Eletrônica** (DAELN). Disciplina: **Fundamentos de Programação 2 - IF62C**/Turma: **S11/S12**. **Prof**: Jean M. **Simão**. Curso: **Engenharia Industrial Elétrica, ênfase Eletrônica/Telecomunicações**. **Prova sobre linguagem C++ – Prova da 1ª Parcial.**

Nome do Aluno: _____ **Turma**: _____
Horário de Começo: _____ Horário de Fim: _____

Leia toda a prova antes de começar, pois os enunciados estão completados uns nos outros.

(Questão 1) Em um programa C++ (para *console*), além de uma classe *Principal_Jogo*, crie também uma classe abstrata chamada de *Personagem*. Esta classe terá um atributo *id*, bem como atributos *x* e *y*.

Isto feito, crie uma classe *Come-come* e uma classe *Fantasma*, ambas derivadas de *Personagem* e cada qual com uma função-membro chamada "*void informe_papel()*". No caso de *Come-come* tal função-membro informará (em tela) "*jogador*", enquanto que no caso de *Fantasma* informará "*inimigo*". Ainda, crie pelo menos um atributo pertinente e respectivos métodos para cada uma destas duas classes.

(Questão - 2) A classe *Come-come* terá um atributo chamado *num_vidas* que deverá (nos seus objetos) ser acessível de alguma maneira, por algum método apropriado.

Ainda com relação à classe *Come-come*, o operador de fluxo (<<) deverá ser sobrecarregado para suportar objetos dela, de maneira que cada objeto de *Come-come* possa ter o valor de seus atributos (incluindo os herdados) impressos em tela.

Por fim, a classe *Come-come* terá o operador de igual (==) sobrecarregado, de maneira que cada objeto de *Come-come* seja capaz de verificar se tem o mesmo valor de *x* e *y* de dado objeto de *Fantasma* passado (como parâmetro por referência) a este operador sobrecarregado.

(Questão - 3) Crie uma classe gabarito (*template*) chamada *Elemento*. Ainda, baseado nesta classe gabarito, crie outra classe gabarito *Lista*, de maneira tal a permitir (subseqüentemente) instanciar lista(s) simplesmente encadeada(s) relativa(s) a ponteiros de objetos de uma dada classe (em cada lista) .

(Questão - 4) Crie uma classe *Lista_Personagens* para ponteiros de objetos de subclasses de *Personagem*. Cada objeto (potencial) da classe *Lista_Personagens* pode ter um número variável de elementos inclusos. Salienta-se ainda que cada objeto de subclasse de *Personagem* pode (potencialmente) participar de diversos objetos de *Lista_Personagens*. Assim sendo, utilize a classe Lista<Tipo> (previamente criada na questão 3) na implementação desta classe *Lista_Personagens*.

Obs.: (a) A classe *Lista_ Personagens* deverá ter um método para listar bem como outro para incluir.
(b) A listagem apresentará o valor de cada atributo de cada objeto, bem como o seu 'papel' no jogo.

(Questão - 5)
- Faça com que a classe *Principal_Jogo* tenha (um atributo privado que será) um objeto da classe *Lista_ Personagens* chamado *obj_Lista_P_1*.
- A classe *Principal_Jogo* deverá ter também um método para listar os objetos de subclasses de *Personagem*, bem como outro método para incluí-los. Estes métodos chamarão métodos do objeto *obj_Lista_P_1* passando os parâmetros necessários (quando necessários).
- A classe *Principal_Jogo* deverá ter ainda um método menu que, em um laço de repetição, permitirá (por meio de alocação dinâmica de memória): (1) Criar Come-come(s); (2) Criar Fantasma(s); (3) Listar Personagens; ou (4) Sair.

Obs. gerais:

(a) Utilizar *cout/cin* para entrada/saída, bem como *string* em vez de vetores de caractere ordinários.

(b) Utilizar os bons princípios da orientação a objetos (nesta questão, bem como em toda a prova).

A interpretação faz parte do conteúdo da prova! Inclua comentários (se for o caso) para deixar claras suas decisões.