Data: 02/10/2018 – 2º Semestre de 2018.
Prof. Dr. Jean M. Simão - DAINF/UTFPR Curitiba-Central.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ - Campus de Curitiba Central – **Dep. Acadêmico de Informática** (DAINF). Disciplina: **Técnicas de Programação – CSE20**.
Prova sobre linguagem C++ / Diagrama de Classes (*UML*) / Orientação a Objetos - 1ª Parcial

Nome do(a) Discente: _____ **Turma**: S73
Curso: *B.S.I.* **Horário de Começo**: _____ **Horário de Fim**: _____

Leia toda a prova, pois os enunciados estão completados uns nos outros.
Utilize os bons princípios de projeto e programação orientada a objetos.
Em tempo, questões de mesmo peso, idem para os tópicos dentro delas (quando houver).

(Questão - 1) (a) Em um programa C++ (*console*) para representação & ´simulação´ de insetos, crie uma classe *Inseto* com um método virtual puro chamado *sobreviver*. **(b)** A classe *Inseto* também permitirá duas classes derivadas dela, nomeadamente *Mosca* e *Lagarta*, sendo que esta ainda é derivada em *Borboleta*. **(c)** *Mosca* terá um atributo booleano chamado *fome* (cujo valor de inicialização é *true*), *Lagarta* terá um atributo inteiro *tempo* (cujo valor de inicialização é *zero*) e *Borboleta* terá como atributos um inteiro chamado *x* e um inteiro chamado *y* inicializados com valor zero. **(d)** Ainda, a classe *Borboleta* terá um atributo estático inteiro chamado *quantidade* (naturalmente inicializado com *zero*) com respectivo método estático *getQuantidade()*.

(Questão - 2) (a) A classe *Mosca* acessará o valor de *quantidade* de *Borboleta* por meio de uma chamada estática de método apropriado, bem como terá um método *sobreviver* que permite saciar fome se o atributo estático *quantidade* supramencionado for maior que uma constante *Max* dada. **(b)** A classe *Lagarta* terá um método *sobreviver* que permite incrementar *tempo* em um (1) se o seu valor for nulo ou positivo. **(c)** A classe *Lagarta* também terá um método chamado *metamorfosear* que permite a cada objeto de *Lagarta*, se o *tempo* nele for maior que *Max*, criar um novo objeto da classe *Borboleta* a luz dele mesmo e depois atribuir -1 a *tempo* **(d)** A classe *Borboleta* terá um método sobreviver que a permite incrementar seu *x* em 10 (simulando assim seu voo).

(Questão - 3) (a) No contexto do programa C++ em questão, crie uma classe chamada *ListaInsetos*, que será uma lista simplesmente encadeada para apontamento de instância da *Inseto*. A *ListaInsetos* deve permitir cada apontamento de instância de *Inseto* ser tratado por um objeto (dinamicamente criado) de uma classe aninhada *ElementoInseto*. **(b)** Essa classe *ListaInsetos* deve ter um método de inclusão. **(c)** Essa classe *ListaInsetos* deve ter um método de *percorrimento* que permite navegar em cada inseto apontado, do último inserido para o primeiro, chamando o método *sobreviver* de cada qual. **(d)** Por fim, tal classe deve ter um método limpar (a ser chamado na destrutora) que desaloca cada objeto ali apontado enquanto inseto, bem como cada objeto de *ElementoInseto*. OBS. **Caso a questão seja feita de outra forma, por exemplo via STL, ela valerá no máximo a metade.**

(Questão - 4) (a) Crie uma classe *Principal* a ser instanciada na função *main*. **(b)** Em tal classe, via construtora, agregue trezentos objetos de *Mosca* e duzentos objetos de *Lagarta*, sendo tais agregações via alocação de memória. **(c)** Nesta classe *Principal* agregue um objeto de *ListaInsetos* chamado *Habitat* que permita registrar todos apontamentos de objetos de *Mosca*, *Lagarta* e *Borboleta* criados no âmbito do sistema em questão como um todo. **(d)** Em um método *executar* dessa classe *Principal*, crie um laço de repetição com 20 iterações, chamando em cada iteração o método de *percorrimento* de *Habitat*.

(Questão - 5) (a) No tocante as questões anteriores, crie apenas *gets* e *sets* pertinentes (i.e., que sejam, utilizados). **(b)** Tal qual, crie construtoras e destrutoras apropriadas. **(c)** Faça *Max* ser um atributo constante estático na classe *Inseto*. **(d)** Primeiramente, antes das demais questões, elabore um diagrama em *UML* das classes solicitadas nelas e de seus relacionamentos, sem necessariamente definir atributos e métodos.