Data: 07/05/2019 – 1º Semestre de 2019.
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 1ª Parcial – Linguagem C++ / Diagrama de Classes (*UML*) / Orientação a Objetos.**

Nome do(a) Discente: _____ **Turma**: S71
Curso: ***Eng. Comp.*** **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++ (para *console*), crie uma classe chamada de *Ponto* com dois atributos protegidos inteiros chamados *x e y*. **(B)** Esta classe terá também um método "*virtual void imprimir ( ) { desenhar(x, y,'.'); }*". **Obs**.: Em tempo, considere a função *void desenhar(int xx, int yy, char cc)* como dada e pronta. **(C)** Crie uma classe *Linha_Ortogonal* derivada de *Ponto* e que agrega objetos de *Ponto*. Esta classe terá atributos inteiros chamados *xf* e *yf*, bem como pelo menos um método, o qual deve ser "*void imprimir()*". **(D)** Tal método permitirá (em tela) desenhar uma linha ou semirreta ortogonal (ou seja, "deitada" ou "de pé"), formada por objetos de *Ponto*, a partir de *x* e *y* como coordenadas de início, bem como *xf* e *yf* como coordenadas de fim. **Obs**.: considere que *x*, *y*, *xf* e *yf* terão valores apropriados para formar linha "deitada" ou linha "de pé".

(Questão - 2) (A) Crie uma classe *Retângulo* derivada de *Ponto*. Esta classe terá, além do *x* e *y* derivados (que servem de coordenada de origem), um atributo inteiro *largura* e outro *comprimento* (ambos sempre positivos). **(B)** Neste sentido, a classe em questão terá ainda quatro atributos do tipo *Linha_Ortogonal* instanciadas com base naqueles atributos. **(C)** Ainda, a classe *Retângulo* terá um método chamado "*void imprimir()*", que permitirá (em tela) desenhar um retângulo a partir do desenho das quatro linhas que a compõem. **(D)** Por fim, esta classe terá o operador de igual, ou seja *bool operator==(Retângulo &)*, sobrecarregado para que cada objeto de *Retângulo* possa verificar se está ou não coincidindo exatamente com outro retângulo.

(Questão - 3) (A) Crie uma classe *ConjPontos* para ponteiros de objetos de *Ponto* por assim dizer. Na prática, a classe *ConjPontos* servirá para apontar para objetos de subclasses de *Ponto*, apontados como *Ponto* bem entendido. **(B)** A classe *ConjPontos* deve ser baseada em alocação e desalocação de memória via comandos *new* e *delete*. **(C)** Ainda, a classe *ConjPontos* deverá ter um método *void incluir (Ponto* p)* que não aceitará mais que mil inclusões. **(D)** Por fim, a classe *ConjPonto* terá outro método *void percorrer()*, o qual fará cada objeto apontado chamar seu método *imprimir()*. **Obs**.: Caso a questão tenha seu item *(B)* feito por outra técnica, ela valerá no máximo metade da nota. Entretanto, isso sim viabilizaria prosseguir com demais questões.

(Questão - 4) (A) Crie uma classe *Principal* que será instanciada na função *main()* do programa. Esta classe terá dois atributos estáticos inteiros chamados de X e Y, inicializando-os com zero e que podem ou não ser incrementados a cada vez que eles são usados. Ainda ele terá dois atributos constantes estáticos LARG e COMP com valores 3 e 6 **(B)** Na classe *Principal* devem ser criados três objetos de *Retângulo* cujos os valores iniciais advêm dos atributos estáticos mencionados. **(C)** Ainda, a classe *Principal* terá método para verificar se há objetos de *Retângulo* geometricamente totalmente coincidentes via operador de igualdade pertinente. **(D)** Outrossim, a classe *Principal* terá um objeto de *ConjPontos*, no qual os endereços destes objetos de *Retângulo* serão registrados, bem como terá um método *void executar()* que permitirá desenhar tais objetos a partir de si.

(Questão - 5) (A) Primeiramente, antes das demais questões, elabore um diagrama em *UML* das classes solicitadas nelas e de seus relacionamentos. **(B)** Defina os principais atributos e métodos das classes no diagrama. Ainda, no diagrama e no código, *set/get* basta fazer em *Ponto* e indicar se há nas demais classes. **(C)** Explique se há caso de polimorfismo previsto nas questões e onde (se for o caso). **(D)** Cite ao menos outras duas técnicas que poderiam ser usadas no item *(B)* da *questão 3*.