Universidade Federal da Fronteira Sul
Ciência da Computação
Programação I - Prof. Fernando Bevilacqua

Prova 2

Nome: _____ **Data:** 19/06/2015

1. A prova pode ser feita a lápis, porém o professor se dará ao direito de não aceitar reclamações relativas à correção.
2. Início da prova 7h30, término 10h00. Manter celulares desligados!
3. Coloque o seu nome nas folhas de resposta.
4. A compreensão das questões faz parte da prova.

Questões

Boa prova!

Você foi encarregado de fazer a modelagem e arquitetura de um jogo. Durante o levantamento de requisitos, você descobriu o seguinte. Todas as entidades que existem (ou existirão) no software precisam ter uma posição no espaço, então elas possuem uma coordenada contendo valores (x, y, z). Existem dois tipos principais de entidades, uma que descreve atores (que são humanoides, ex.: pessoa) e outra que descreve construções (ex.: casa, prédio, etc.). Independente do tipo da entidade, todas elas possuem um valor inteiro que descreve quantos pontos de vida a entidade possui. Além disso, todas elas possuem uma funcionalidade que calcula o seu volume, porém o volume de um ator e o volume de uma construção são calculados de forma diferente. Não é possível determinar o volume de uma entidade por si só, é necessário saber se ela é um ator ou uma construção. Em relação aos atores, você constatou que existem dois tipos deles: o personagem e o NPC. Diferentemente do NPC, o personagem possui N itens (ex. espada, escudo, etc), que por sua vez são descritos por um nome e um custo em ouro. Um NPC possui um nome, uma profissão e uma quantidade de ouro. Em relação às construções, existe o mercado e o quartel. Qualquer construção possui um valor inteiro que descreve o seu tempo de vida (há quanto tempo ela existe). O mercado possui um saldo em ouro referente às vendas feitas, já o quartel possui um contador que indica quantos soldados foram treinados até o momento naquele quartel. Algumas das entidades descritas podem ser danificadas por ações externas, como é o caso do personagem e do mercado. Danificar algo significa descontar um certo valor da quantidade de vida da entidade em questão. Algumas entidades podem ser reparadas, como é o caso do personagem e de todas as construções. Reparar algo significa adicionar um certo valor à vida da entidade em questão. Por fim, você descobriu que o programa pode ter um número infinito de entidades, porém todas devem ser guardadas num mesmo vetor na memória do programa.

1)(3.0) Faça o diagrama UML que descreva a modelagem desse sistema.

2)(2.5) Implemente o código Java da classe `Curanderia`, que é uma construção. Essa classe é responsável por reparar entidades do jogo. Ela possui dois métodos. O método `repara` recebe um vetor de entidades e ele adiciona 5 pontos de vida a todas as entidades reparáveis existentes no vetor. Além disso, ele altera a posição (x, y, z) de todas as entidade para origem (0, 0, 0). O método `quantoPodemPagar` recebe um vetor de entidades e ele retorna a quantidade de ouro equivalente à soma do custo de todos os itens que os personagens existentes no vetor passado como parâmetro possuem.

3)(1.5) Você foi informado que existem as seguintes classes de exceções já implementadas: `EErro` (filha de `Exception`), `EErroEntidade` (filha de `EErro`), `EEntidadeMorta` (filha de `EErroEntidade`), `EEntidadeSemPosicao` (filha de `EErroEntidade`), `EErroConstrucao` (filha de `EErro`) e `EErroConstrucaoDestruida` (filha de `EErroConstrucao`). Assuma que existe uma classe chamada `Gerenciador` com um método estático chamado `teste`, que levanta todas as exceções citadas.

a. Como seria a assinatura (protótipo) desse método `teste`?
b. Faça um pequeno programa que use o método `teste`, pegando todas as exceções que podem ser levantadas (para cada exceção pega, imprima na tela o nome dessa exceção).

4)(3.0) Você foi encarregado de implementar uma nova interface que permita que entidades possam colidir umas com as outras. A interface deve se chamar `Colidivel`. A interface deverá conter um método chamado `colidiu`, que recebe como parâmetro qualquer elemento colidível (ou seja, que também implementa a interface `Colidivel`). O método retorna `true` se o objeto que está executando o código colidiu com o objeto passado como parâmetro, ou `false` caso contrário. Implemente o código Java dessa interface. Você recebeu a ordem de fazer com que qualquer elemento do sistema que possua posição (x, y, z) possa colidir, tanto uns com os outros quanto entre si. Assuma que um elemento colide com outro se os valores (x, y, z) de ambos são iguais. Faça o código em Java necessário para cumprir essa tarefa que você recebeu. Não é necessário escrever o código de todos os métodos e propriedades das classes, apenas o método que implementa as obrigações da interface.