UNIVERSIDADE FEDERAL DE PELOTAS
CENTRO DE DESENVOLVIMENTO TECNOLÓGICO
CURSOS DE CIÊNCIA E ENGENHARIA DE COMPUTAÇÃO
Disciplina de Programação Orientada a Objetos - 2019/1
Prof.: Felipe de Souza Marques (felipem@inf.ufpel.edu.br)

Especificação do Trabalho Final

1. Objetivo

Exercitar os conceitos de Programação Orientada a Objetos, focando em abstração, encapsulamento, herança e polimorfismo.

2. Avaliação

A nota do trabalho totalizará 10 pontos, que são distribuídos nos seguintes itens de avaliação:

a) (3 pontos) Habilidade em estruturar programas pela decomposição da tarefa em subtarefas, utilizando subprogramação para implementá-las.
b) (3,5 pontos) Emprego dos conceitos de POO: abstração, encapsulamento, herança e polimorfismo.
c) (3,5 pontos) Atendimento aos requisitos mínimos da especificação do problema.

Uma pontuação extra será atribuída aos alunos que completarem os seguintes requisitos:

a) (+ 5%) Utilização de arquivo(s) para fazer a configuração inicial (ou definição do cenário inicial) do problema abordado.
b) (+ 10%) Utilização de Padrões de Projeto.
c) (+ 20%) Construção de uma interface gráfica (GUI).

3. Entrega do trabalho

O trabalho deve ser entregue via AVA até o dia 08/07/2019 até as 12h e pode ser feito em grupos de até 3 pessoas. **Os trabalhos devem ser apresentados para o professor até às 18:50 do dia 08/07/2019** em sala de aula.

Um representante do grupo deve submeter os seguintes arquivos para completar a entrega do trabalho:

a) Diagrama de Classes: um arquivo pdf contendo o diagrama de classes final do projeto;
b) Arquivo JAR: o arquivo JAR contendo o projeto Java compilado (existem diversos tutoriais de como fazer isso na internet – se ficarem com dúvida, procurem o professor).
c) Readme: Arquivo texto contendo as instruções básicas para o uso correto do programa enviado.
d) Arquivos fonte: para simplificar, pode-se compactar o diretório do projeto em um arquivo ZIP.

Este conjunto de arquivos deve ser **compactado no formato ZIP** em **um único arquivo** e enviado para entrega via página da disciplina no AVA.

4. Código de Honra

O trabalho deve ser implementado na sua totalidade, sem uso de código de outros (colegas ou não). O trabalho enviado deve representar um esforço honesto em resolver o problema - isto é, não é algo "pela metade", que não implementa funcionalidades essenciais. Violações a esta conduta serão penalizadas e o violador não só terá nota nula neste trabalho, mas também não terá direito a enviar os próximos trabalhos, ficando portanto sem parte da nota da disciplina. Plágios estão sujeitos a sanções administrativas pelo colegiado do curso.

5. Proposta de Trabalho: Jogo Baseado na Ideia do World-of-Zuul

5.1. Introdução

A ideia do jogo World-of-Zuul é simples e foi explorada em nossas aulas práticas. Até o momento temos a possibilidade de criar um mapa de salas, inserindo itens e personagens em cada uma delas. A partir deste cenário inicial, um personagem herói pode se deslocar no mapa para coletar itens valiosos e combater inimigos.

Agora é o momento de você exercitar sua criatividade e criar a sua história para o jogo! Com base em um conjunto de regras mínimas pré-estabelecidas, você poderá criar a sua história e aumentar o conjunto de regras, incluindo novos procedimentos, novos itens, novos tipos de personagem, criar tuneis secretos de uma sala para outra mais distante, etc.

5.2. Requisitos Mínimos

a) **O mapa** do jogo deve conter pelo menos **12 salas**. Algumas destas salas podem estar completamente vazias, servindo apenas de corredor de acesso à outras salas. As demais, devem conter itens e/ou inimigos de acordo com a história do jogo.

b) Ao menos um **personagem principal** deve existir: o nosso herói. Além dos atributos já existentes no projeto desenvolvido em aula, ele deve considerar um atributo para guardar um número de “moedas” (que pode ser um valor inteiro). Cada conjunto de 1000 moedas ocupa uma unidade de peso no inventário do herói. Valores intermediários serão sempre arredondados para o próximo valor inteiro, ou seja, se o herói estiver carregando 1200 moedas, elas estarão ocupando 2 unidades de peso no inventário do herói. Também deve ser permitido que o herói se equipe com um ou mais equipamentos que garantam características especiais. Por exemplo, armas que aumentam o dano causado em combate, ou ainda, um escudo e/ou armadura que previna dano em combate.

c) **Deve haver no mínimo dois tipos de inimigos**: **comuns e chefes**. Todos inimigos devem tem um atributo que informa a quantidade de

moedas que ele carrega. Ao eliminar um inimigo, ele deve ser removido da sala e as moedas que ele carregava devem ser adicionadas ao inventário de itens da sala. O herói poderá coletar as moedas do inimigo derrotado a partir do mapa de itens da sala. Os inimigos do tipo chefe podem carregar um ou mais itens que concedem habilidades a quem os carrega/equipa. Por exemplo, um item que aumenta o dano causado de um ponto de energia para dois, ou ainda, um item que aumenta a energia em X pontos (existem várias outras possibilidades). Ao eliminar o inimigo chefe, todos os itens devem ser movidos para o inventário da sala. O herói pode se apossar dos itens a partir do mapa de itens da sala.

d) Alterar o sistema de luta implementado durante as aulas práticas, de modo que se considere pontos de danos e multiplicadores a partir de itens que os personagens carregam. Para um sistema mais sofisticado é interessante considerar que os personagens possuem pontos de ataque e defesa. Toda vez que se profere um ataque, existe uma probabilidade de se acertar o golpe. Exemplos podem ser encontrados em: https://centrorpg.com/index.php?topic=13160.0 ou http://rpg-swordartonlinerpg.weebly.com/sistema-de-danodefesa.html .
e) Deve-se prever **itens** de diferentes tipos:
    I. Instantâneo: por exemplo, um alimento contém uma determinada quantidade de energia que, quando comidos pelo herói, converte a energia do alimento em energia extra para o herói (observe o limite de energia do herói). Este tipo de item pode ser carregado pelo herói (no inventário) e utilizado quando for necessário. O exemplo do alimento é um dos itens obrigatórios que devem constar no jogo.
    II. Permanentes: por exemplo, enquanto o herói carregar um item do tipo permanente, ele aumenta um determinado atributo do herói: energia máxima, carga de peso máximo ou quantidade de dano causado em cada ataque realizado. Neste caso, esta seria uma habilidade passiva. Armas, escudos, elmos, anéis, etc são considerados itens permanentes. Itens permanentes devem ter um atributo de durabilidade. Toda vez que o item é utilizado ou o herói sofre algum dano, os itens que estão equipados também sofrem perda de pontos de durabilidade. Estes pontos poderão ser restaurados a partir do uso de outros itens ou outros personagens com habilidades para consertar itens.

Seja criativo!!! A partir destes requisitos, proponha o seu jogo! Para aqueles que quiserem se aventurar em uma interface gráfica mais elaborada (isso **não** é um requisito mínimo), recomento o uso do framework slick2D (http://slick.ninjacave.com). Ele permite a criação de mapas, uso de imagens para

personagens, sons etc. Tem vários tutorias disponíveis na internet. Outro framework bastante interessante e mais completo é o LibGDX - https://libgdx.badlogicgames.com/.

Anexo I

Especificações iniciais feitas nas aulas práticas

1. A especificação do primeiro exercício prático está no AVA:
2. Especificação do dia 13/05:
    a. Na classe Game: inserir um atributo do tipo personagem para representar o herói;
    b. Na classe Room:
        i. Acrescentar um novo mapa para armazenar referências aos personagens da sala: map<String, Personagem>; (a chave refere-se ao nome do personagem)
        ii. No método que gera a descrição da sala: acrescentar lista de personagens da sala.
    c. Acrescentar o comando "attack", para realizar o ataque do herói sobre o vilão (passando o nome do mesmo);
        i. Colocar comando no Enum CommandWords;
        ii. Modificar o método processCommand da classe Game para tratar o comando de ataque;
        iii. Quando o inimigo morrer, retirar ele do mapa de personagens da sala.
3. Especificação do dia 20/05:
    a. Criar a classe Item. Ela deve conter os seguintes atributos:
        i. Nome do item (String);
        ii. Peso do item (int).
    b. Na classe Heroi, instanciar um mapa para controlar o inventário do herói:
        i. map<String, Item>; (a chave se refere ao nome o item)
        ii. Adicionar o atributo 'limiteDePeso' (int), para estabelecer o peso máximo que o herói pode carregar;
        iii. Adicionar métodos para inserir e remover itens do inventário;
            - boolean inserir(Item item): insere itens respeitando o limite máximo de peso;
            - Item remover(String nome): remove o item do inventário e o retorna;
            - Opcional: criar método auxiliar 'pesoAtual' para retornar o somatório dos pesos de todos os itens do inventário.