UNIVERSIDADE FEDERAL DE LAVRAS

Práticas de Programação Orientada a Objetos

GAC106 – 2022-1

Exercícios: World of Zuul

Versão: 2022-06-30

Prof. Julio César Alves *DAC - ICET* *https://julio.dac.ufla.br*

É hora de colocarmos a **mão na massa** mais uma vez.

Os passos 1 e 2 apenas repetem o que já foi colocado como atividade nos slides 2.4 - Design de Classes.

Os passos 3 em diante são novidades.

Passo 1 – Definindo o tema do jogo

Obs.: este é o mesmo passo já apresentado no Slide 11 da apresentação sobre Design de Classes.

O código do World of Zuul será usado para entendermos os conceitos de design de classes. Mas o jogo fica mais interessante se for criado por você mesmo! Nesta atividade você deve mudar o tema do jogo para um criado por você. Pode ser qualquer jogo que tenha como estrutura de base um jogador que se move por locais diferentes (nas direções norte, sul, leste e oeste). Faremos os passos abaixo na aula presencial.

No papel, você deverá seguir os seguintes passos:

- Definir um tema para seu jogo.
- Definir um nome para o jogo.
- Planejar um mapa (com pelo menos cinco ambientes).
- Definir uma forma do jogador ganhar (ex: coletar um item mágico ou salvar uma pessoa, etc.).

Depois, no computador você deve alterar o código do World of Zuul da seguinte forma:

- Alterar o nome do projeto para o nome do seu jogo.
- Alterar os métodos *imprimirBoasVindas* e *imprimirAjuda* para se adequar à temática do seu jogo.
- Alterar o método *criarAmbientes* para criar o mapa que você planejou para seu jogo.
- Teste seu jogo (por enquanto o jogador conseguirá apenas se movimentar no mapa).

Passo 2 – Aplicando as melhorias das aulas

Durante as aulas, foram feitas várias alterações no código que melhoraram o design de classes, evitando problemas como o alto acoplamento, a baixa coesão ou a duplicação de código, por exemplo.

Esta atividade é para te lembrar que você precisa ter feito essas alterações a partir do código de exemplo World of Zuul Bad disponível no Campus Virtual.

Em resumo as alterações são:

- Remoção da duplicação de código (Slides 19 e 20).
- Remoção de atributos públicos e uso da solução com *HashMap* (Slides 23 a 27).
- Alteração para permitir que o jogador possa se mover para cima e para baixo (Slide 27).
- Criação do método getDescricaoLonga para melhorar o design baseado em responsabilidade (Slides 28 a 30).
- Removido o acoplamento implícito e criado o comando *observar* no jogo (Slides 32 a 37).

Passo 3 – Criando itens nos ambientes

Nosso jogo por enquanto não permite fazer muita coisa. Mas, e se quiséssemos acrescentar itens que os jogadores pudessem coletar e usar?

Vamos neste passo permitir que nossos ambientes possuam itens. Para isso, faça o seguinte:

- Crie uma classe para representar itens:
  - Com os atributos: nome, peso e descrição.
  - Os ítens que podem existir no seu jogo vão depender muito da temática que você usou.
    - Logo, a classe precisa se chamar *Item*; ela deve ter o nome que faça sentido pra você.
- Altere a classe Ambiente para que ela tenha:
  - Um atributo item.
  - Um segundo construtor que receba um item.
  - Um método que retorna um booleano indicando se o ambiente tem um item.
  - Altere o método getDescricaoLonga para incluir a informação sobre um item de um ambiente.
    - Se o ambiente tiver um item, incluir o nome e a descrição do item.
    - Se não tiver, deve exibir uma mensagem como “Não há nada aqui”.
- Altere a classe Jogo:
  - Altere a criação de ambientes para que pelo menos dois deles tenham algum item.
    - Os itens precisam ser diferentes em cada ambiente.
  - O jogo pode ficar mais interessante se o método *irParaAmbiente* exibir apenas a descrição do novo ambiente (e não a descrição longa). Isso fará com que o comando “observar” seja mais útil.

Não se esqueça de testar o seu jogo com as alterações feitas!

Passo 4 – Coletando itens

Vamos agora alterar nosso jogo para que o jogador consiga pegar um item do ambiente. Para isso:

- Crie uma palavra de comando chamada “pegar” (na classe PalavrasComando);
- Crie dois métodos na classe Ambiente:
  - Um para consultar o item existente (retorna null se não houver item).
  - Outro para coletar o item do ambiente, ou seja, ele deve deixar o atributo item do ambiente com valor null e retornar o item (dica: use uma variável auxiliar).
- Trate a palavra de comando “pegar” na classe Jogo.
  - Se o usuário digitar apenas “pegar”, dê uma mensagem apropriada (ex: “Pegar o que?”).
  - Se o usuário digitar o nome do item que está no ambiente:
    - O item deve ser coletado do ambiente, e uma mensagem deve ser exibida dizendo que o usuário pegou tal item.
    - Obs: por enquanto não é necessário guardar o item que o jogador pegou.
  - Se o item não estiver no ambiente (ou for uma palavra que não existe):
    - Deve ser exibida uma mensagem indicando que não há esse item no ambiente.

Teste suas alterações!

Passo 5 – Carregando itens

Vamos agora permitir que o jogador carregue itens que depois possam ser usados em outros lugares para realizar ações (tais como: abrir portas, desbloquear passagens, destruir monstros, etc). Nosso código fica melhor se o **refatorarmos**, criando uma classe para representar o jogador.

Faça os seguintes passos:

- Crie uma classe para representar o jogador:
  - Dê um nome para a classe que tenha a ver o tema do seu jogo.
  - Um jogador deve ter um nome e uma lista de itens que estão sendo carregados.
    - Poderia ser uma mochila ou algo que faça sentido com o tema do seu jogo.
  - Crie o construtor que recebe o nome do jogador e cria a lista de itens vazia.
  - Crie um método para retornar o nome do jogador.
  - Crie um método para adicionar um item na lista do jogador.
  - Crie um método que, a partir do nome do ítem, remove-o da lista do jogador e o retorne.
  - Crie um método que retorne uma única string informando todos os itens que o jogador está carregando.
- Crie um atributo jogador na classe `Jogo` e o inicialize adequadamente.
- Altere o tratamento do comando “pegar” na classe `Jogo`.
  - Quando o jogador pegar um item que está no ambiente, esse item deve ser adicionado na lista de itens do jogador.
- Altere o comando “observar” para que, além de fazer o que já faz, ele também exiba os itens que estão sendo carregados pelo jogador.

Jogue e teste suas alterações!

[Opcional] Passo 6 – Bloqueando ambientes

Vamos tratar neste passo o bloqueio dos ambientes de uma forma bem simples. A ideia é que apenas uma das saídas de um ambiente possa estar bloqueada (ou nenhuma delas).

Para isso faça as seguintes alterações na classe Ambiente:

- Crie um atributo string que indique qual saída (direção) está bloqueada.
  - Se o atributo tiver valor null, indicará que não há nenhuma saída bloqueada.
- Crie um atributo string que indique o nome do item que desbloqueia a saída bloqueada.
  - O atributo terá valor null se o ambiente não tiver nenhuma saída bloqueada.
- Crie um método ajustarSaidaBloqueada que recebe uma direção, um ambiente e um nome de item de desbloqueio.
  - Tal método chamará o ajustarSaida para a direção e o ambiente e, em seguida, guardará a direção passada como a saída bloqueada e o nome do item passado como o item para desbloqueio.
- O método getSaida deve ser alterado de forma que:
  - Se a direção passada for de uma saída bloqueada, retorne null.
  - Se for para uma saída não bloqueada, retorne o ambiente como já fazia antes.

Em seguida, na criação dos ambientes na classe Jogo, bloqueie uma das saídas. Ou seja, substitua alguma chamada do ajustarSaida de algum ambiente por uma chamada do ajustarSaidaBloqueada, indicando o nome do item de desbloqueio.

Teste seu jogo!

Se tudo estiver certo você não conseguirá passar pela saída bloqueada. No próximo passo veremos como usar um item para desbloquear a saída.

[Opcional] Passo 7 – Usando os itens

Agora vamos realmente tornar os itens úteis, usando-os para desbloquear as saídas dos ambientes. A ideia é que o jogador possa digitar um comando "usar algo", onde *algo* é o nome de um item que ele está carregando e que será usado no ambiente. Se o item for usado em um ambiente com saída bloqueada, e for o item que desbloqueia aquela saída, a mesma será desbloqueada.

Para isso faça as seguintes alterações no código:

- Crie o comando "usar" na classe PalavrasComando.
- Na classe Ambiente, crie um método que permita ao jogador usar um item.
  - Tal método deve receber, por parâmetro, o objeto item a ser utilizado.
  - Se o item recebido tem o mesmo nome do item para desbloqueio, o atributo de saída bloqueada recebe o valor null.
  - O método deve retornar um booleano indicando se uma saída foi desbloqueada ou não.
- Na classe do jogador, crie um método que recebe uma string com o nome de um item e retorne se o jogador tem aquele item.
- Faça o tratamento do comando "usar" na classe Jogo:
  - Se o usuário não digitar o nome do item a ser usado dê uma mensagem apropriada.
  - Se o usuário digitar o nome de um item:
    - Que o jogador não tem: informe o usuário.
    - Que o jogador tem: chame o método de usar o item da classe Ambiente e, caso uma saída tenha sido desbloqueada, remova o item da lista de itens do jogador.
    - Dê uma mensagem para o usuário caso ele tenha desbloqueado a passagem.
- Na criação dos ambientes na classe Jogo, configure os ambientes de forma que o jogador tenha que ir em um ambiente pegar um item, para desbloquear a saída de algum outro ambiente.

Teste seu jogo! Você deve agora conseguir desbloquear a passagem que o jogador não tinha acesso :)