MAC-212 — Laboratório de Computação

Exercício Programa 1: Aventura![*]

Prazo: 25 de abril

1 Introdução

Nosso objetivo é construir um sistema de execução de jogos tipo *adventure*, como apresentado em classe. Neste tipo de jogo o personagem principal (aventureiro) executa ordens dadas pelo jogador. O sistema descreve o resultado de cada ação do aventureiro, sempre em forma de texto.

A aventura acontece em um mundo virtual composto de lugares interligados por passagens, transferências mágicas, teletransporte, etc. Os objetos espalhados por esse mundo podem interagir com o aventureiro de diversas formas. Jogos desse tipo estão disponíveis em diversos lugares:

- De dentro do emacs, execute `M-x dunnet`
- No Linux, rode o programa `adventure`
- Outra possibilidade no Linux é o `battlestar`
- Um jogo "clássico", *The Hitchhikers Guide to the Galaxy*, pode ser jogado via Internet, em `http://www.douglasadams.com/creations/infocomjava.html`.

O projeto será desenvolvido em Java, em equipes de uma ou duas pessoas. Caso o projeto seja feito em dupla, todos os detalhes da solução devem ser discutidos e entendidos pelos dois integrantes da equipe. Não deixe seu colega de equipe fazer tudo sozinho!

2 O Mundo

O mundo é composto por lugares e objetos (coisas) que serão implementados por objetos Java. Para evitar e confusão entre objetos do mundo virtual e objetos Java (que podem ou não corresponder a objetos do mundo virtual), usarei o termo "coisa" para um objeto do mundo virtual. Uma "coisa" pode ser um objeto inanimado ou um ser vivo qualquer.

Tanto os lugares como as coisas possuem um nome e uma descrição.

[*]Boa parte do enunciado deste exercício-programa foi tomada de um exercício similar, criado pelo Prof. Marco Dimas Gubitoso. Obrigado, Gubi!

2.1 Lugares

Um lugar possui as seguintes características:

- Nome — é usado internamente pelo sistema para identificar o lugar. Não pode haver mais de um lugar com o mesmo nome.
- Descrição curta — para que o usuário saiba onde está. É uma palavra ou expressão usada como "nome externo" do lugar. Mais de um lugar pode ter o mesma descrição curta (as salas de um labirinto, por exemplo).
- Descrição — texto que descreve a parte fixa do lugar, que não é afetada pelos comandos do usuário. Esta parte fixa e invariante pode ser trocada, em princípio, por alguma ação especial dentro do jogo.
- Saídas — um `Map` (tabela com duas colunas) indicando as passagens para outros lugares. Cada entrada desse tabela contém uma direção e o nome do lugar alcançado seguindo nessa direção.
- Coisas — uma tabela com as coisas existentes no lugar. Ela pode ser organizada como a tabela de saídas.
- Propriedades — são valores genéricos que permitem especificar melhor o estado do lugar, de acordo com as necessidades do jogo.
- Ações — tabela com os verbos que têm significado especial neste lugar.

A ação "olhar", que dá a descrição de um lugar, pode ser aplicada a todos os lugares. Ela é executada automaticamente quando um lugar for visitado pela primeira vez.

2.2 Coisas

Coisas são muito parecidas com lugares, mas existem algumas diferenças importantes. As coisas possuem as seguintes características básicas:

- Nome — identifica a coisa de maneira única. É usado internamente pelo sistema e serve também como "nome externo" visto pelo usuário. Não pode haver mais de uma coisa com o mesmo nome.
- Descrição — texto que descreve a coisa. Só aparece se o usuário pedir explicitamente.
- Localização — diz onde a coisa se encontra. Pode ser um lugar, uma coisa que pode conter outras (vide abaixo), ou o próprio aventureiro.
- Propriedades — são valores genéricos que permitem especificar melhor as características da coisa, de acordo com as necessidades do jogo. Algumas propriedades podem ser tamanho, peso ou cor. Além disso, algumas coisas podem ter a capacidade de conter outras (mochila, mala, vaso, sacola, etc).
- Ações — tabela com os verbos que tem significado especial para esta coisa.

Uma propriedade especial que pode ser atribuída a uma coisa é a animação: a cada movimento do aventureiro, um função de atualização é chamada para alterar algumas das propriedades do coisa (lugar, parte da descrição, etc).

As seguintes ações podem ser aplicadas a todos as coisas:

- Examinar – descreve a coisa.
- Pegar — passa a coisa para o aventureiro, se possível (podem haver restrições quanto ao número de coisas, tamanho ou peso que o aventureiro pode carregar).
- Largar — a coisa (que precisa estar com o aventureiro) é colocada no lugar onde o aventureiro se encontra. Com argumentos adicionais pode-se colocar uma coisa dentro de outra, desde que a segunda tenha capacidade de conter outras e espaço livre suficiente.

Ações específicas para algumas coisas podem ser "ligar", "destruir", "esfregar" , etc. Procure ter uma implementação padrão para verbos mais gerais.

2.3 O Aventureiro

O aventureiro é um caso muito especial de coisa animada. Ele pode conter outras coisas, se as estiver carregando ou vestindo, e recebe atualizações diretamente do usuário. É claro que ele pode receber outras atualizações, efetuadas automaticamente pelo sistema, de propriedades como cansaço, fome e energia.

2.4 Tabelas

Para facilitar a implementação e tornar a mecânica do jogo mais rápida, é interessante manter algumas tabelas globais. Eis alguns exemplos:

- o mundo — pode ser implementado como um `Map` entre nomes de lugares e objetos Java que implementam os lugares.
- coisas animadas — a cada passo o conjunto de coisas animadas é percorrido para atualizar cada uma delas.
- verbos — todo pode estar num `Map` global que associa cada verbo ao tratamento *default* do verbo. Nas instâncias dos lugares e das coisas ficam apenas os tratamentos dos verbos que tem significado específico para o lugar ou coisa.

3 Um "Empurrão Inicial"

Uma "solução (muito) incompleta" para este exercício está disponível na página de MAC–212. Nela você encontra as classes `br.usp.ime.mac212.adventure.Place` (uma implementação inicial dos lugares), `br.usp.ime.mac212.adventure.Main` (com uma versão bem simples de programa principal) e `br.usp.ime.mac212.util.Console` (usada para leitura de dados do console).

4 Recomendações

- Sua primeira tarefa é estudar e rodar a "solução incompleta". Entenda-a completamente antes de começar a fazer o EP.

- Note que falta *muita* coisa na "solução incompleta". Nela não existem coisas, ações, animação... Não há um objeto que implemente o aventureiro. E os lugares só estão parcialmente implementados (entre outras coisas, falta a descrição curta).

- O modo como a "solução incompleta" inicializa o mundo virtual (método `createWorld` da classe `br.usp.ime.mac212.adventure.Main`) é tedioso e inconveniente no caso de um mundo com um grande número de lugares e de coisas. Numa fase posterior veremos uma maneira melhor de se fazer isso. Por isso não gaste tempo criando um mundo enorme neste EP. O objetivo agora é montar uma infra-estrutura flexível que suporte bem as características dos jogos tipo *adventure*. É claro que para testar sua infra-estrutura você vai precisar de um conjunto variado de lugares e de coisas, mas o tamanho do mundo não é importante por si só.

- Não saia escrevendo código sem pensar! Invista algum tempo planejando sua implementação antes de escrevê-la. Decida que classes você vai precisar, que informações cada classe precisa manter em seus campos, de que forma suas classes vão interagir umas com as outras, etc.

- Lembre sempre que todo programa deve ser organizado em funções ou métodos pequenos (30 linhas em média, uma página no máximo[1]) com objetivos muito bem definidos e que se adaptem naturalmente ao problema que se quer resolver. Todo método deve ser precedido de um comentário que diz sucintamente o que o método faz!

- Se um método ficou muito grande e obscuro, reorganize-o! Considere a possibilidade de transferir parte de suas tarefas para métodos auxiliares.

- Se uma classe parece grande e sobrecarregada de responsabilidades (métodos demais, por exemplo), considere a possibilidade de criar classes auxiliares que assumam parte das responsabilidades da classe.

Bom Trabalho!

[1] Esta é uma "regra de bolso", que não pode ser levada ao pé da letra em todas as situações. O exemplo típico de situação que escapa dessa regra é um método contendo um comando `switch` com muitos casos.