Design de classes
Atividades

world-of-zuul

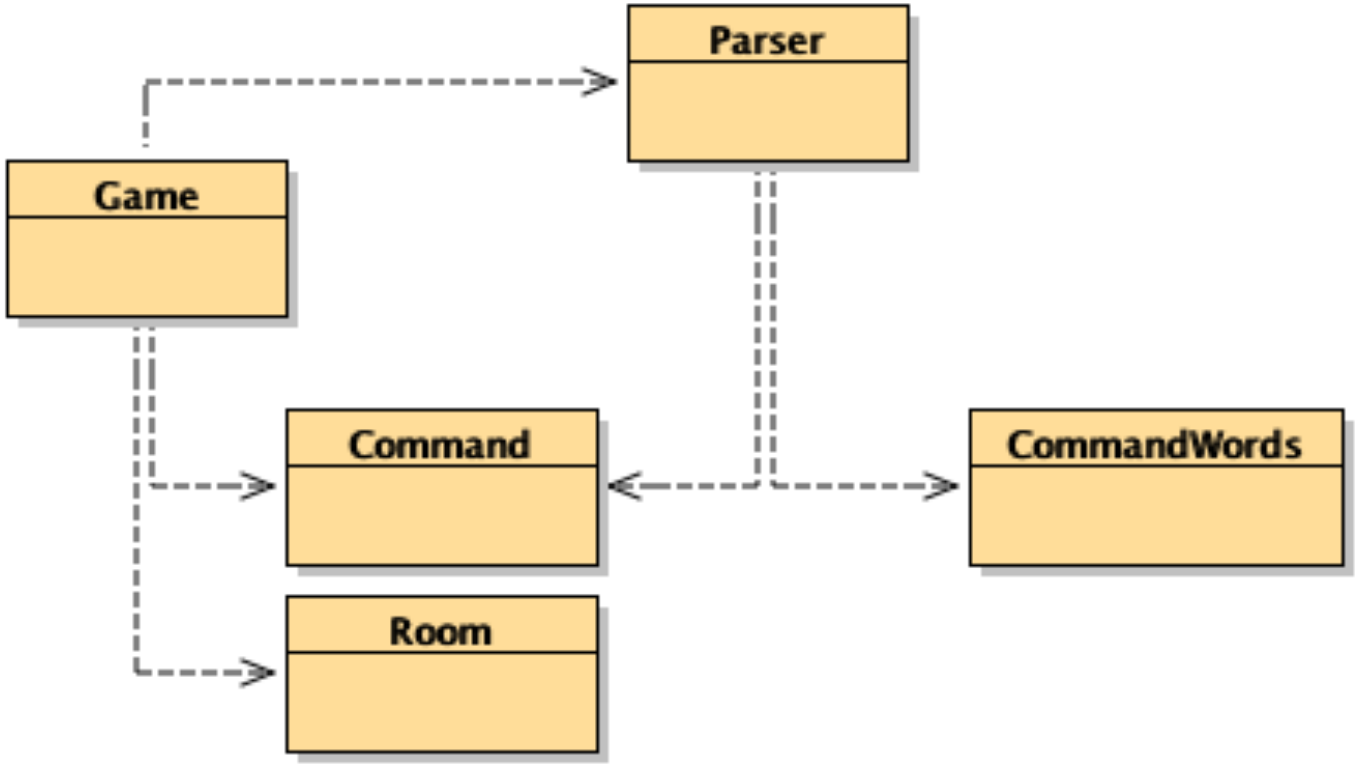
- O jogo world-of-zuul tem como modelo um jogo que envolve a busca de seu caminho por um sistema complexo de cavernas, localizar um tesouro escondido, utilizar palavras secretas e outros mistérios, tudo com o objetivo de marcar o número máximo de pontos

world-of-zuul

- Neste lab utilizaremos uma implementação simples desse jogo de aventura com base em texto. A implementação está incompleta apenas com o estabelecimento de um cenário (e deslocamentos no cenário) onde o jogo possa ser realizado.

Exemplo

- Abra o projeto zuul - Design ruim
  - Execute e explore a aplicação



Exemplo

- Durante a exploração da aplicação, responda às seguintes perguntas:
  - O que essa aplicação faz?
  - Que comandos o jogo aceita?
  - O que cada comando faz?
  - Quantas salas estão no cenário?
  - Desenhe um mapa das salas existentes.

Exemplo

- Depois de saber o que a aplicação faz, tente descobrir o que cada classe individual faz. Você precisa ver o código-fonte (disponível no SIGAA) para fazer isso. Não precisa entender todo o código-fonte. A leitura dos comentários e a cabeçalho de método são suficientes. A seguir uma pequena descrição das classes.

Descrição das classes

- A classe CommandWords define todos os comandos válidos no jogo
  - Usa uma lista de strings para representar as palavras de comando
- Um objeto Command representa um comando que foi inserido pelo usuário
- A classe Parser é responsável por lê a entrada terminal e tenta interpretá-la como comandos
  - Cria objetos Command que representam o comando inserido

Descrição das classes

- Um objeto Room representa uma localização em um jogo. As salas podem ter saídas que levam a outras salas.
- A classe Game configura o jogo e entra em um loop para ler e executar comandos
- A classe Main inicia a aplicação

Atividade 1

- Verificar a possibilidade de **fazer extensões** no projeto, por exemplo, adicionar uma nova direção de movimento (up e down)
- Pelo menos duas classes estão envolvidas nessa alteração: Room e Game
  - Método setExits
  - Método createRoom
  - Método printWelcome
  - Método goRoom

Atividade 2

- Verificar **duplicação de código**
- nos métodos printWelcome e goRoom (classe Game)
- Elaborar um método privado printLocationInfo com as informações sobre a localização atual

Atividade 3

- Utilizar o encapsulamento para reduzir o acoplamento (campos privados com métodos de acesso)
  - Campos privados e utilizar um método de acesso para obtê-los (classe Room)
  - Método getExit na classe Room
    - getExit(self, direction)
  - É preciso alterar também a classe Game