POO – Programação Orientada a Objetos
Trabalho 1
Prof. José Fernando Rodrigues Júnior

- Integrantes do grupo:
  Rafael Miranda Lopes - nºUSP 6520554

- Plataforma de Desenvolvimento:
  Desenvolvido em Java utilizando a IDE NetBeans.

- Instruções de compilação e execução:
  Projeto do NetBeans.

- Diagrama UML simplificado:
  Ver página 2.

- Esclarecimentos:

1. O jogo foi feito sem utilizar a interface sugerida, para que o jogo pudesse ser feito com uma aproximação mais fiel ao jogo original, em termos de mecânica. Foi consultado um tutorial de desenvolvimento de engine em Java na página http://abrindoojogo.com.br/djj-index. As classes do pacote *Core: ImageHandler*, *InputHandler*, *GameSpeedTracker* e *Game* foram retiradas do site ou utilizadas como base, de forma consciente, ou seja: o funcionamento das classes foi estudado e entendido; algumas delas foram alteradas. A classe *InputHandler* apresenta um defeito bem conhecido do *KeyListener*, que por vezes não identifica o *release* de uma tecla; o efeito disso é que, ocasionalmente, o avatar do jogador irá ficar com uma "tecla travada", devendo ser pressionada e solta até que o evento de *release* seja corretamente lido. Foram tomadas algumas medidas para reduzir o problema, mas não de forma completamente satisfatória. As classes *FontHandler* e *SpriteAnimation*, presentes no projeto, não foram utilizadas mas não foram retiradas porque possivelmente serão utilizadas para a segunda fase do trabalho.

2. As 6 fases a serem implementadas foram testadas e têm solução de acordo com o jogo original. Os itens recolhidos têm sua pontuação alterada conforme a ordem em que são coletados. O *top score* é armazenado apenas enquanto o jogo roda. Todos os elementos foram implementados e suas mecânicas estão de acordo com o jogo original.

3. Ao perder todas as vidas, aparece uma tela de Game Over e, ao vencer todas as fases, aparece uma tela de créditos. Em ambos os casos, o jogo é reiniciado após alguns segundos.

4. A classe abstrata solicitada (Elemento) chama-se, aqui, *MapObject*, da qual derivam todos os elementos de jogo presentes no mapa, conforme descrito no diagrama UML.

5. A classe *Fase* solicitada, chama-se, aqui, *GameMap*, que apresenta funções de controle do fluxo de jogo, além de simplesmente armazenar os dados de cada fase. Para trocar de fase, basta reconstruir a *GameMap*, que foi projetada como um *Singleton*, pois há apenas uma fase ativa. Não houve tempo hábil para fazer com que a classe carregasse as fases de um arquivo, para que ficasse mais elegante e fácil de editá-las.

6. As sprites utilizadas, com exceção das de jogador e monstro, foram feitas pelo autor do trabalho.

Diagrama UML