Relatório de Programação Orientada para Objetos 2024/2025

Membros do grupo:

- *Gonçalo Moita, 123283*
- *Erick Ozaki, 122088*

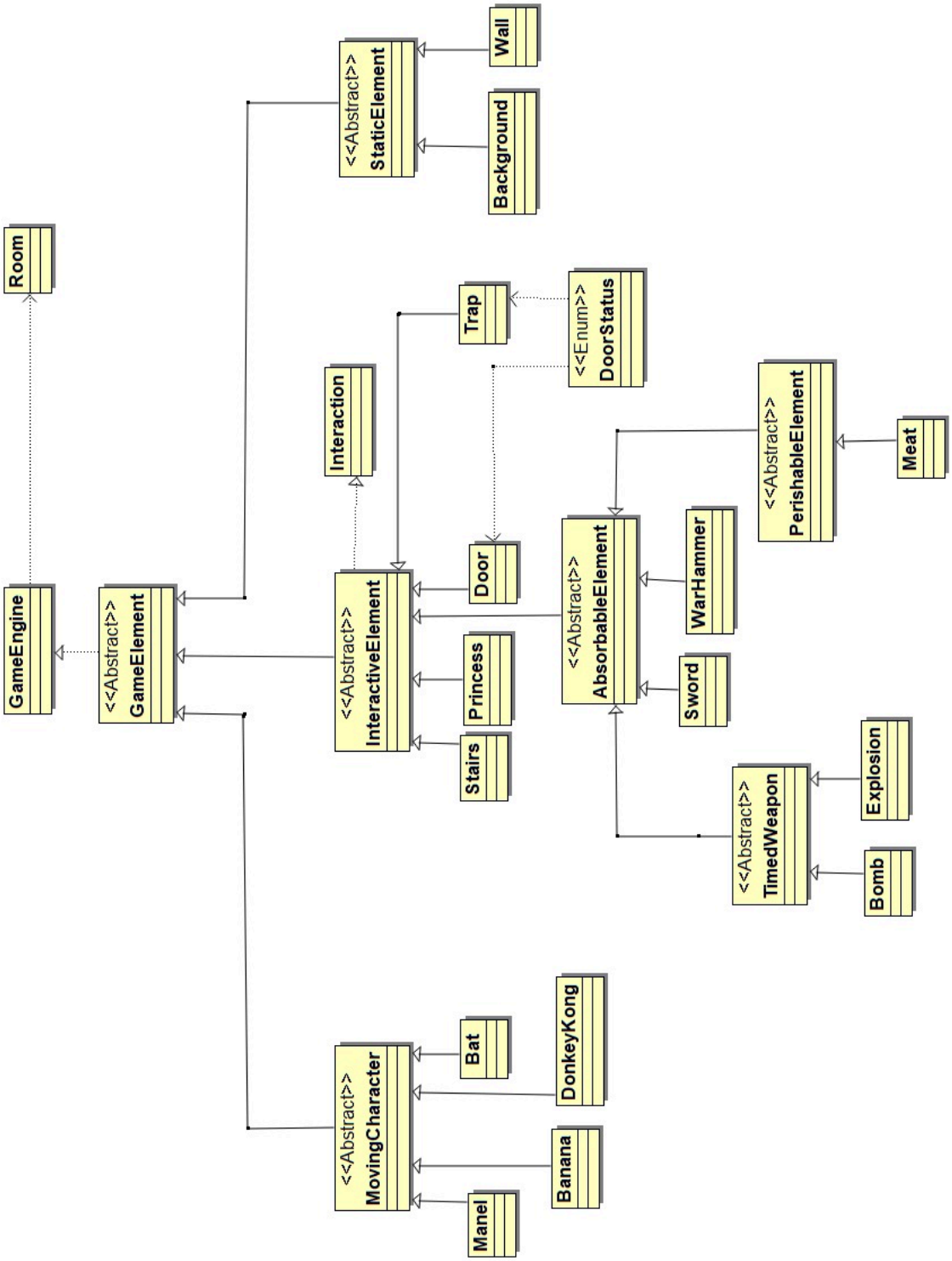
Docente que orientou o trabalho (práticas):

- *Pedro Sousa Romano*

Declaração de honra:

Ao entregar este trabalho e relatório declaramos implicitamente que todo o código entregue é da inteira responsabilidade do grupo e todos os elementos conhecem em profundidade todas as partes do trabalho entregue.

Diagrama UML



Observações:

- O UML refere-se somente a classes, interfaces e demais entidades por nós criadas;
- Algumas das imagens fornecidas para o projeto foram trocadas por outras, por gosto (e.g. `BadMeat`), enquanto outras foram replicadas e/ou alteradas, como é o caso da imagem do herói refletida, para que ele possa caminhar para os dois lados;
- As implementações incluem uso dos diversos recursos discutidos em aula, onde vistos como solução adequada ao problema em mãos: interfaces, herança, classes estáticas, solitões, etc;
- Alvejamos implementações tão sintáticas e genéricas quanto possível, o que se nota, por exemplo, com a implementação única de movimento (na classe `MovingCharacters`), que acaba por servir à movimentação do herói e do inimigo principal, o `DonkeyKong`, de maneira que os dois têm todas as mesmas hipóteses de movimento, exceto por a movimentação do herói ser controlada, enquanto a do inimigo é aleatória;
- Realizamos alguns testes automáticos com ferramenta de nossa própria implementação[1], assemelhada às assinaturas dos métodos JUnit, de modo a detetar erros de maneira rápida e automática;
- Prezamos pela separação de responsabilidades entre as classes, de maneira que nenhum dos personagens e/ou objetos de jogo se comunica com a `ImageGUI` (o que é responsabilidade exclusiva da `GameEngine`), nem tampouco a `GameEngine` controla a movimentação dos `MovingCharacters` (que é executada por eles mesmos, e somente solicitada pela `GameEngine`);
- As classes `Manel`, `Princess` e `ScoreBoard`, são exemplo de implementação do tipo Solitão (Singleton), por sua natureza propor unicidade aos exemplares de suas respetivas manifestações (instâncias);
- Definimos a classe `Player` como uma classe aninhada, devido a só se usar no contexto da classe `ScoreBoard`, e além disso ajuda melhorar a organização do projeto.
- O enumerador `DoorStatus` criado tem como função ajudar na transição de um estado de algumas das classes de objetos, nomeadamente `Door` e `Trap`;
- O interface `Interaction` é um mecanismo que permite que os objetos interajam com o Manel ao ter contacto com ele, mas cada um a sua maneira;

[1] A “ferramenta” (talvez um exagero chamá-la assim) foi desenvolvida para servir como suporte aos testes das atividades de AED no semestre passado. O código fonte original pode ser encontrado em: https://github.com/eozaki/aed2024/blob/main/src/main/java/Utils/Test.java. Também fazem parte do mesmo projeto ficheiros de testes que fazem uso dessa classe de apoio, bem como se fez no presente projeto.