











TRABALHO PARA A DISCIPLINA DE
TÉCNICAS DE PROGRAMAÇÃO DO CURSO
DE ENGENHARIA DE COMPUTAÇÃO
DA UTFPR TURMA S71

Adryan C. Feres e Thales G. B. dos Santos

Sonho++

Tabela de Requisitos

N.	Requisito	Status
1	Apresentar graficamente menu de opções aos usuários do Jogo, no qual pode se escolher fases, ver colocação (<i>ranking</i>) de jogadores e demais opções pertinentes.	
2	Permitir um ou dois jogadores com representação gráfica aos usuários do Jogo, sendo que no último caso seria para que os dois joguem de maneira concomitante	
3	Disponibilizar ao menos duas fases que podem ser jogadas sequencialmente ou selecionadas, via menu, nas quais jogadores tentam neutralizar inimigos por meio de algum artifício e vice-versa	
4	Ter pelo menos três tipos distintos de inimigos, cada qual com sua representação gráfica, sendo que ao menos um dos inimigos deve ser capaz de lançar projétil contra o(s) jogador(es) e um dos inimigos dever ser um 'Chefe'.	

5	Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias e sendo pelo menos 3 instâncias por tipo.	
6	Ter três tipos de obstáculos, cada qual com sua representação gráfica, sendo que ao menos um causa dano em jogador se colidirem	
7	Ter em cada fase ao menos dois tipos de obstáculos com número aleatório de instâncias (<i>i.e.</i> , objetos), sendo pelo menos 3 instâncias por tipo.	
8	Ter em cada fase um cenário de jogo constituído por obstáculos, sendo que parte deles seriam plataformas ou similares, sobre as quais pode haver inimigos e podem subir jogadores.	
9	Gerenciar colisões entre jogador para com inimigos e seus projeteis, bem como entre jogador para com obstáculos. Ainda, todos eles devem sofrer o efeito da gravidade no âmbito deste jogo de plataforma vertical e 2D.	
10	Permitir: (1) salvar nome do usuário, manter/salvar pontuação do jogador (incrementada via neutralização de inimigos) controlado pelo usuário e gerar lista de pontuação (ranking). E (2) Pausar e Salvar Jogada.	

Percentual de Requisitos cumpridos

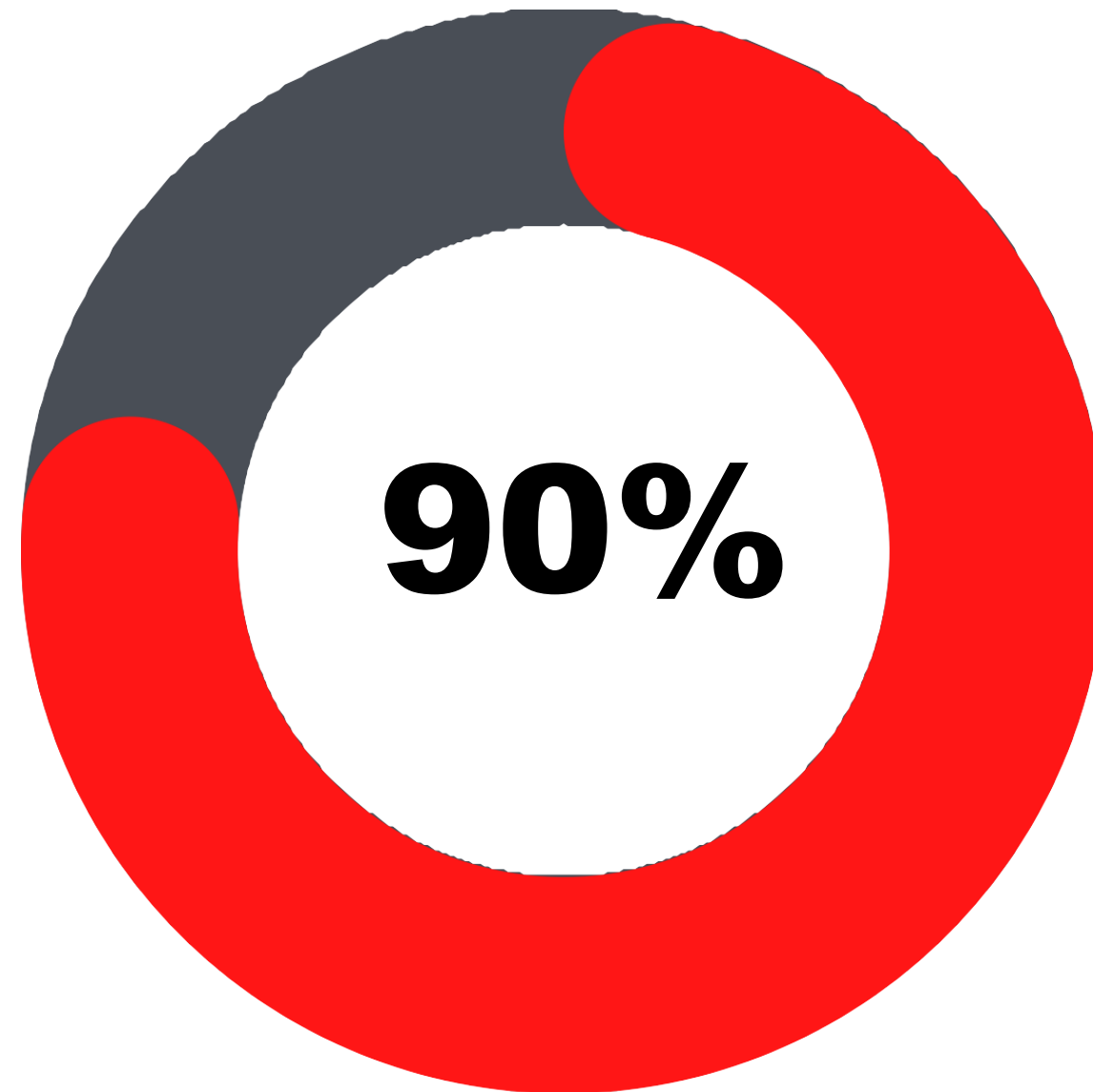
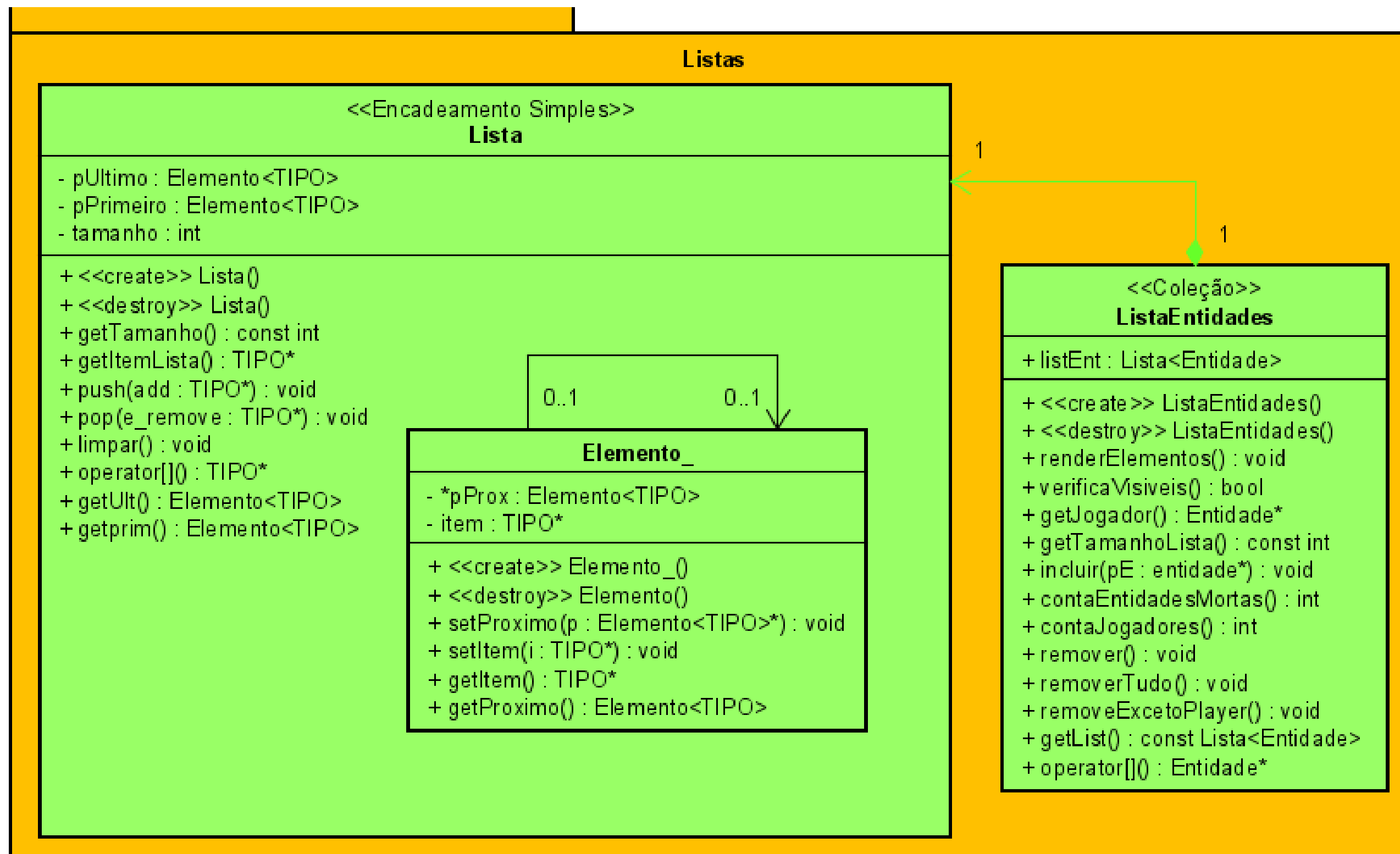
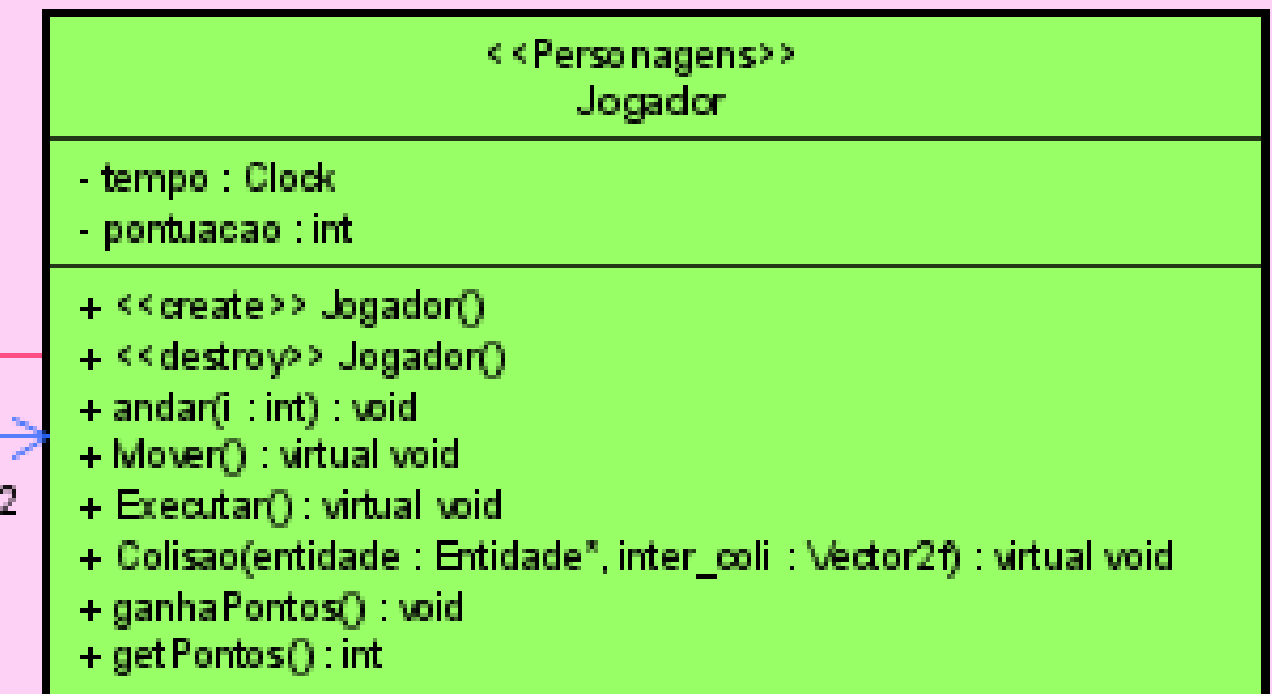


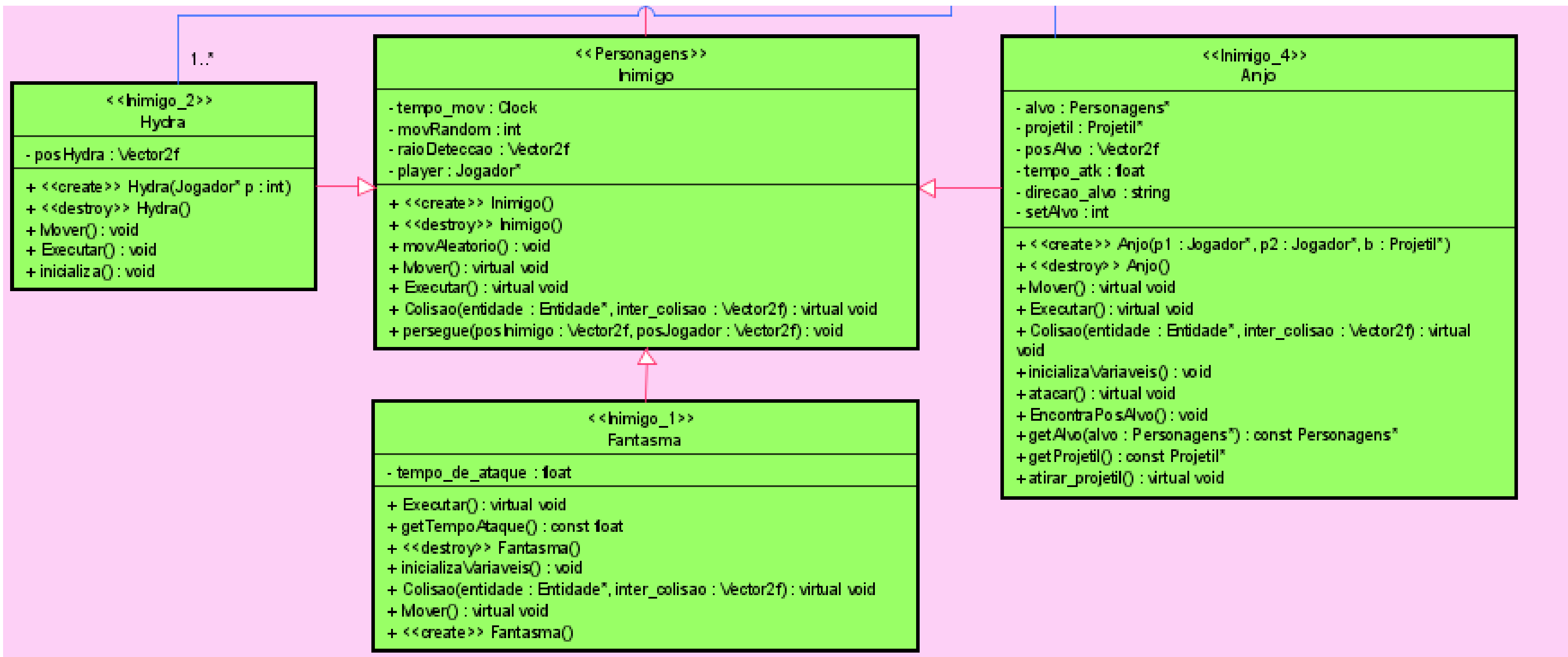
Diagrama de Classes

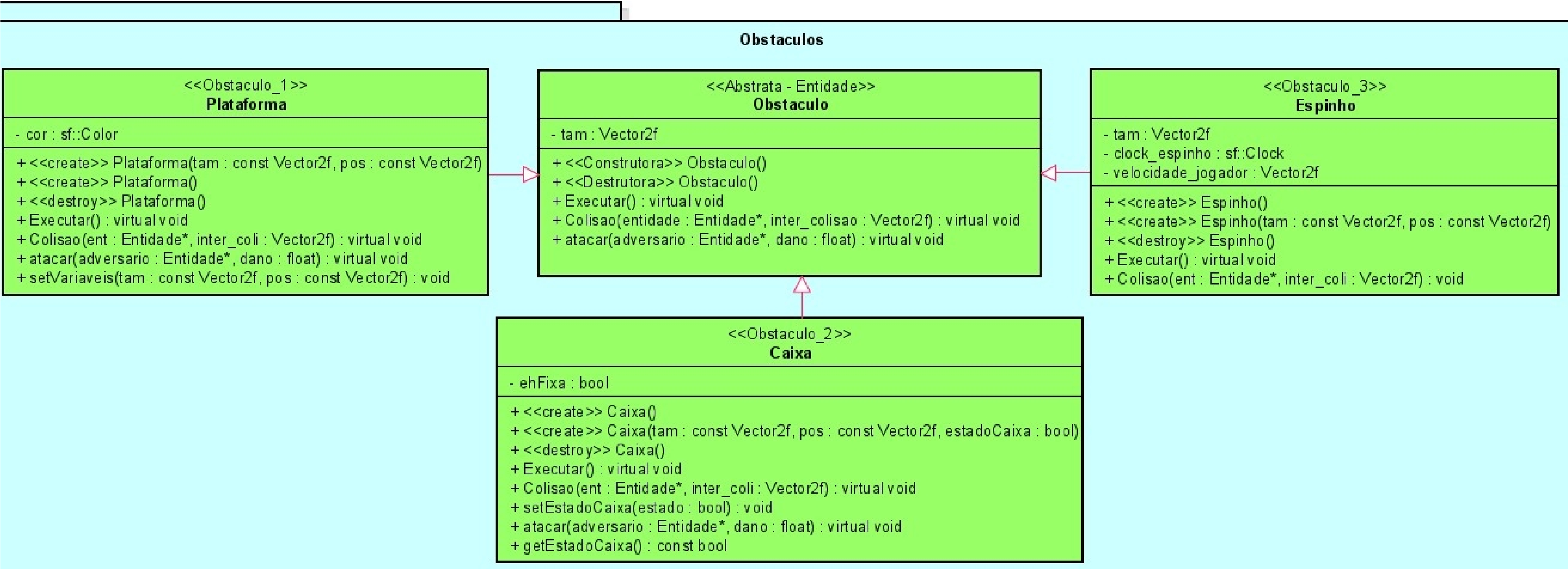


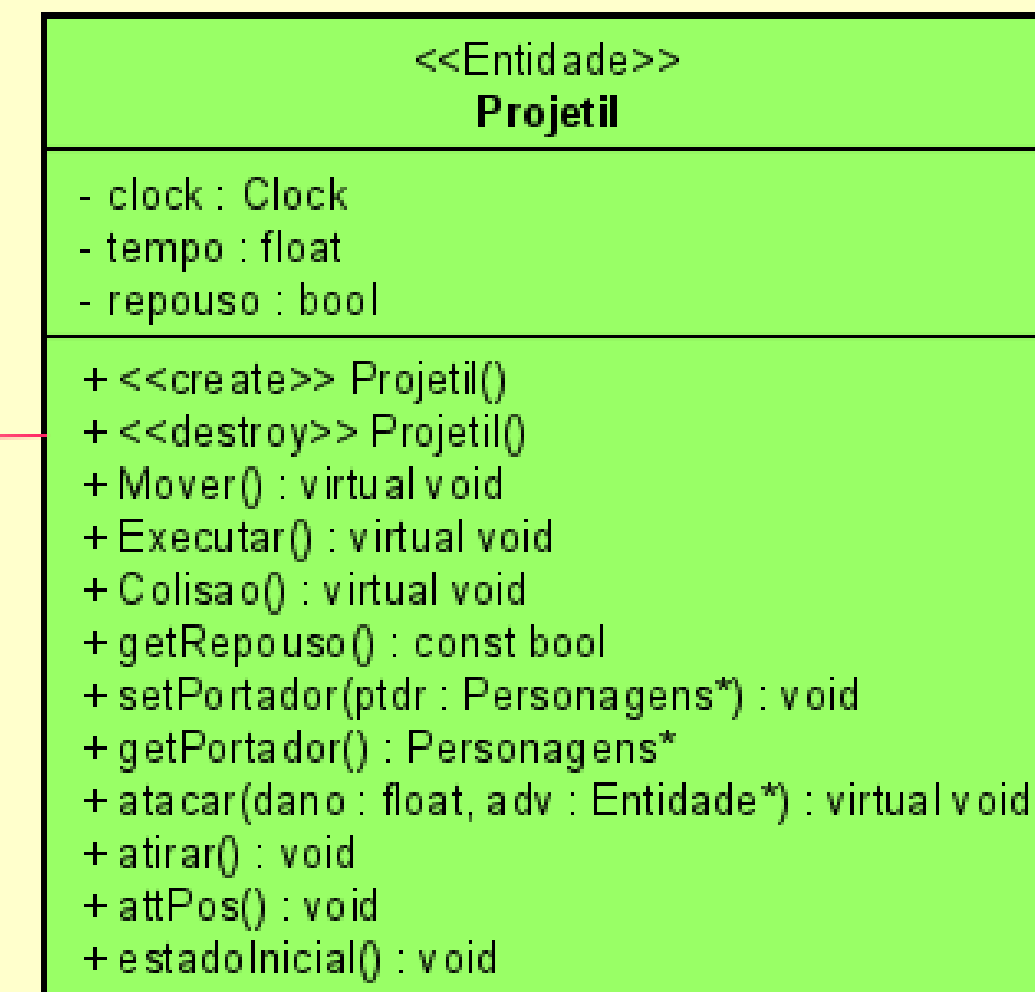
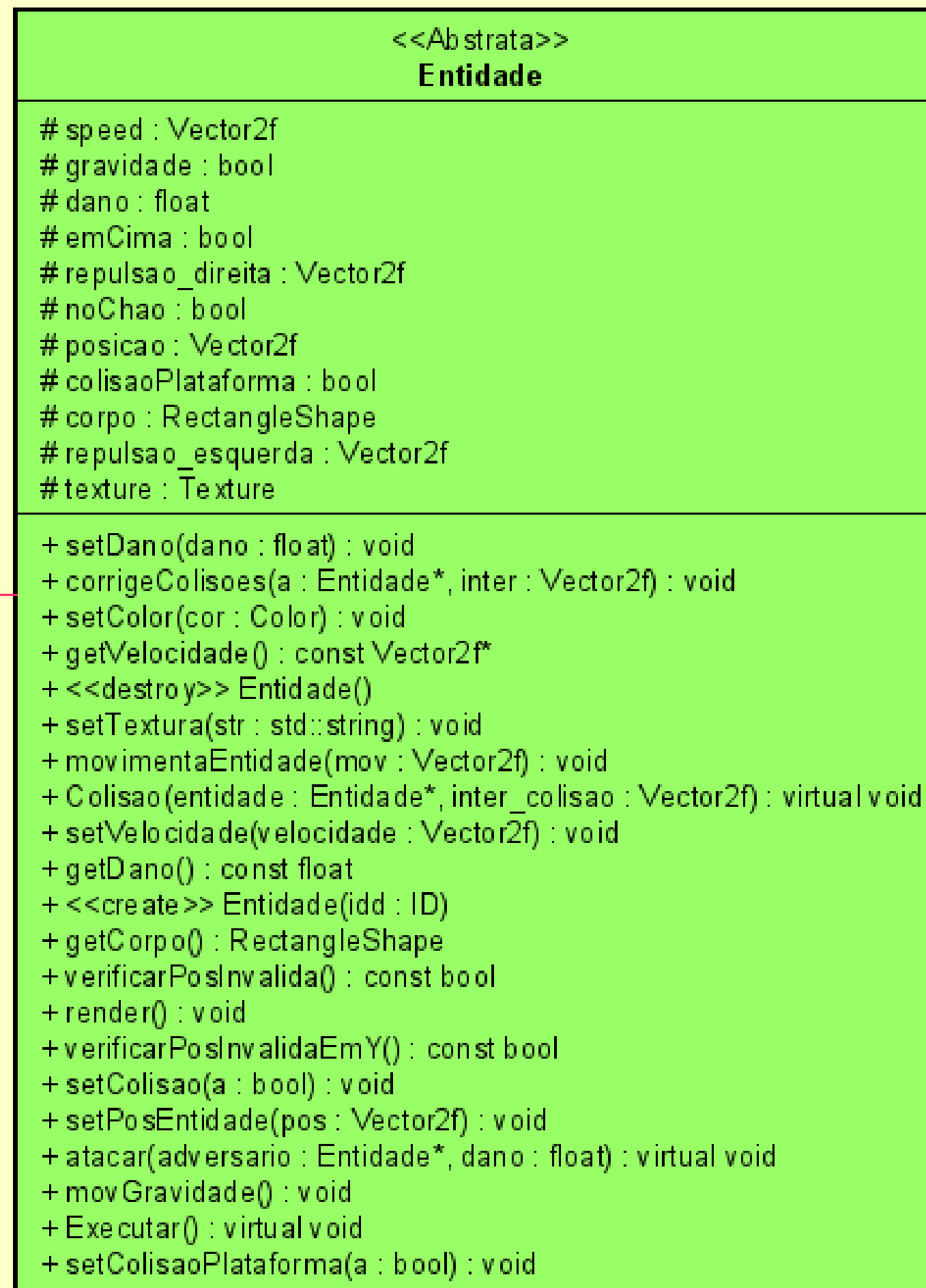
Personagens

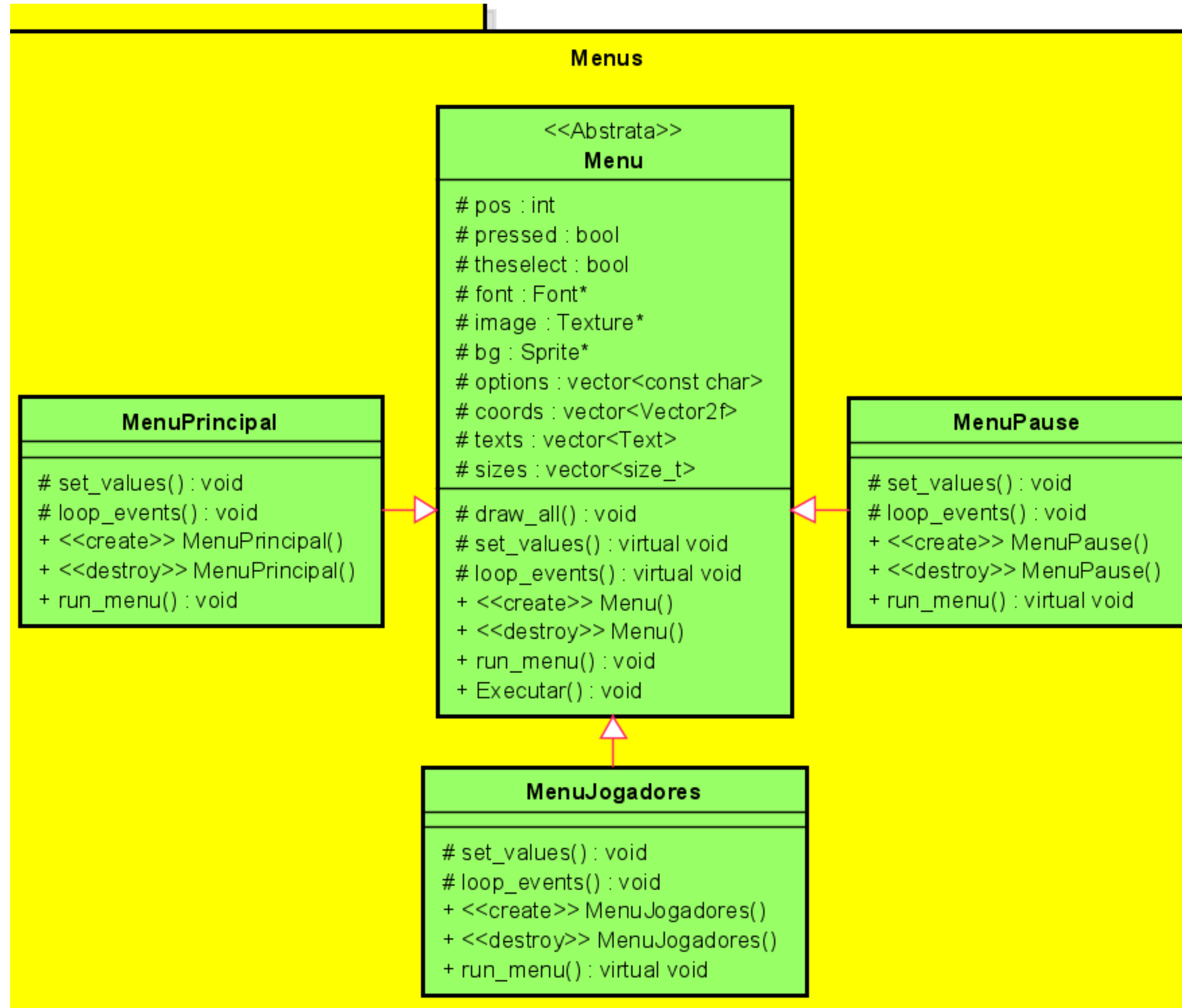


1..2









Gerenciadores

<<Infraestrutura - SFML - Singleton>>

Gerenciador_Grafico

```
- window : RenderWindow*
- Clock : Clock
- pGrafico : GerenciadorGrafico*
- <<create>> GerenciadorGrafico : .
- estado : ID
+ dt : float
- fase1 : bool
- umJogador : bool

+ GerenciadorGrafico()
+ getGerenciadorGrafico() : GerenciadorGrafico*
+ getWindow() : RenderWindow*
+ Limpar() : void
+ DesenhAr(corpo : RectangleShape) : void
+ Mostrar() : void
+ FecharJanela() : void
+ isWindowOpen() : bool
+ atualizaTempo() : void
+ setEstado(id : const ID) : void
+ getEstado() : ID
+ verificaEventos(evento : Event) : const bool
+ getUmJogador() : const bool
+ setUmJogador(bool t : int) : void
+ setFase1(t : bool) : void
+ getFase1() : const bool
```

<<Mediador - STL>>

Gerenciador_Colisoes

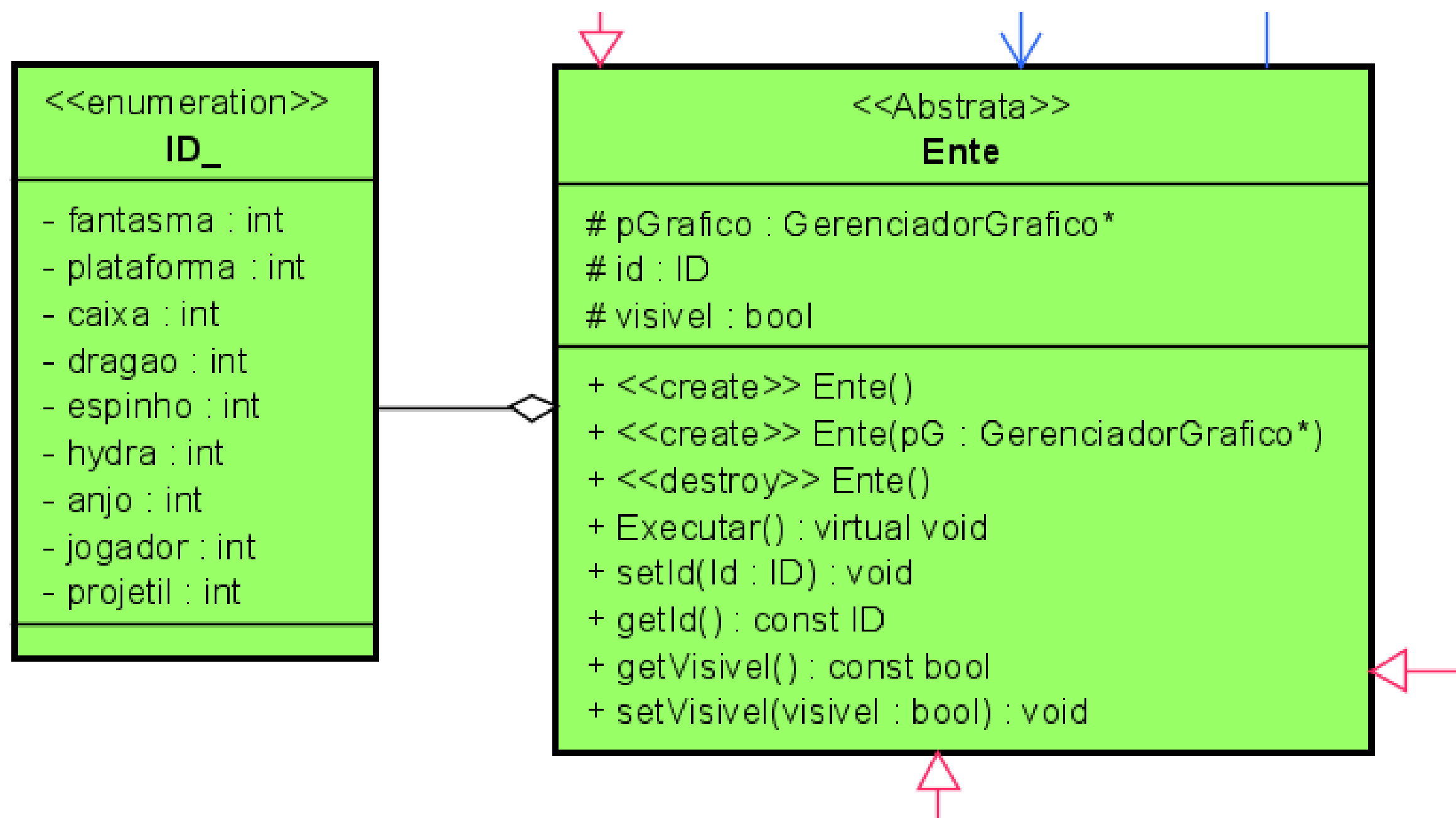
```
- lista_obstaculos : ListaEntidades*
- lista_personagens : ListaEntidades*

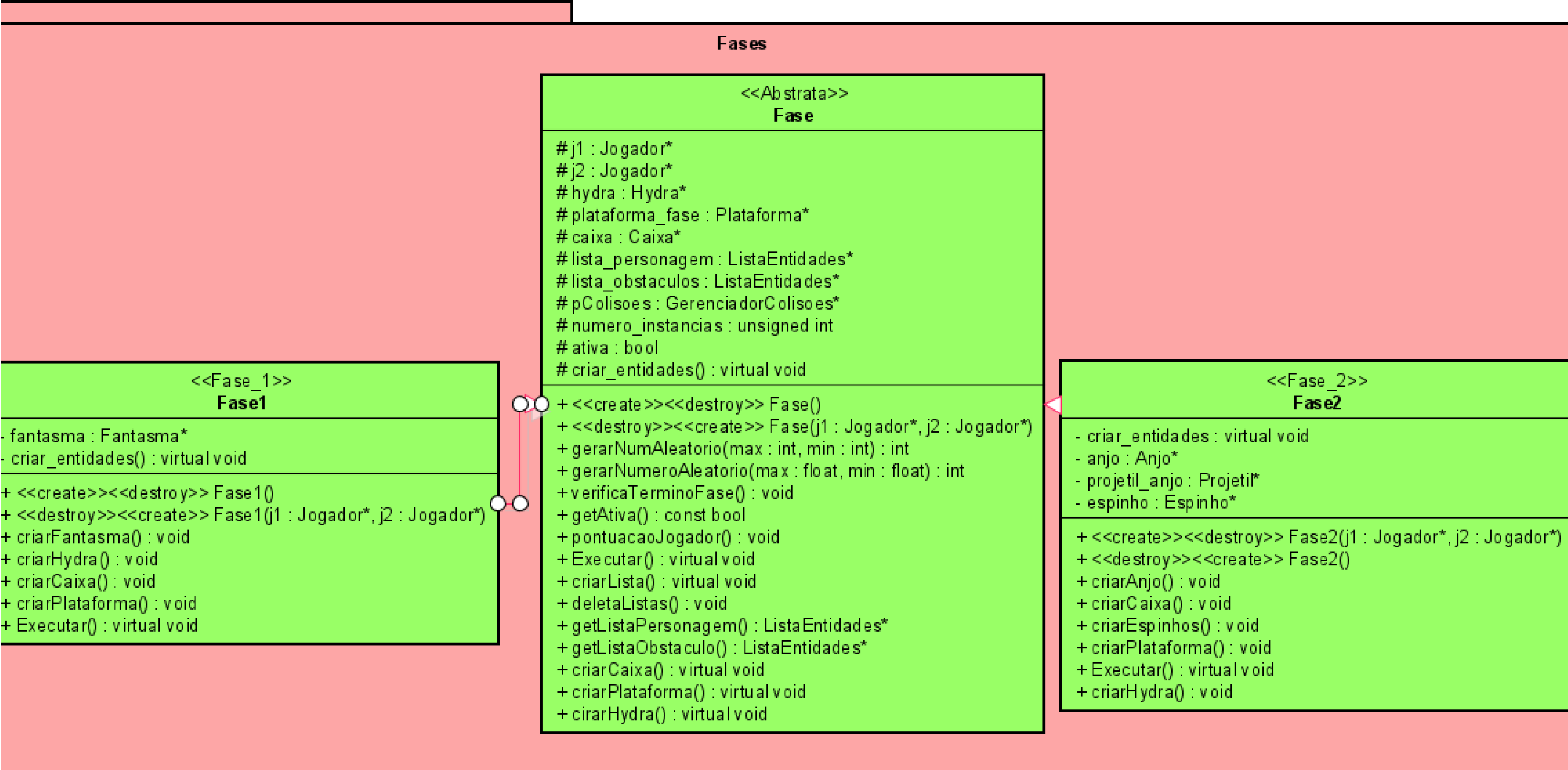
+ <<create>> Gerenciador_Colisoes(l_personagem : ListaEntidades*, l_obstaculo : ListaEntidades*)
+ <<destroy>> Gerenciador_Colisoes()
+ calculaColisoes(ent1 : Entidade*, ent2 : Entidade*) : Vector2f
+ verificaColisoes() : void
+ colisaoPersonagens() : void
+ colisaoPersonagemObstaculos() : void
+ setListas(lper : ListaEntidades*, lobs : ListaEntidades*) : void
+ Executar() : void
+ colisaoObstaculos() : void
```

GerenciadorEvento

```
- pGrafico : GerenciadorGrafico*
- Jogador1 : Jogador*
- Jogador2 : Jogador*
- tecla_pres : int
- pEvento : gerenciadorEvento*
- gerenciadorEvento() : .

+ getGerenciadorEvento() : void
+ setJogador1(j : Jogador*) : void
+ setJogador2(j : Jogador*) : void
+ verificaTeclaPressionada() : void
+ operation29() : void
+ verificaTeclaPressionada(tecla : Keyboard::Key) : void
+ verificaTeclaSolta(tecla : Keyboard::Key) : void
+ Executar() : void
```





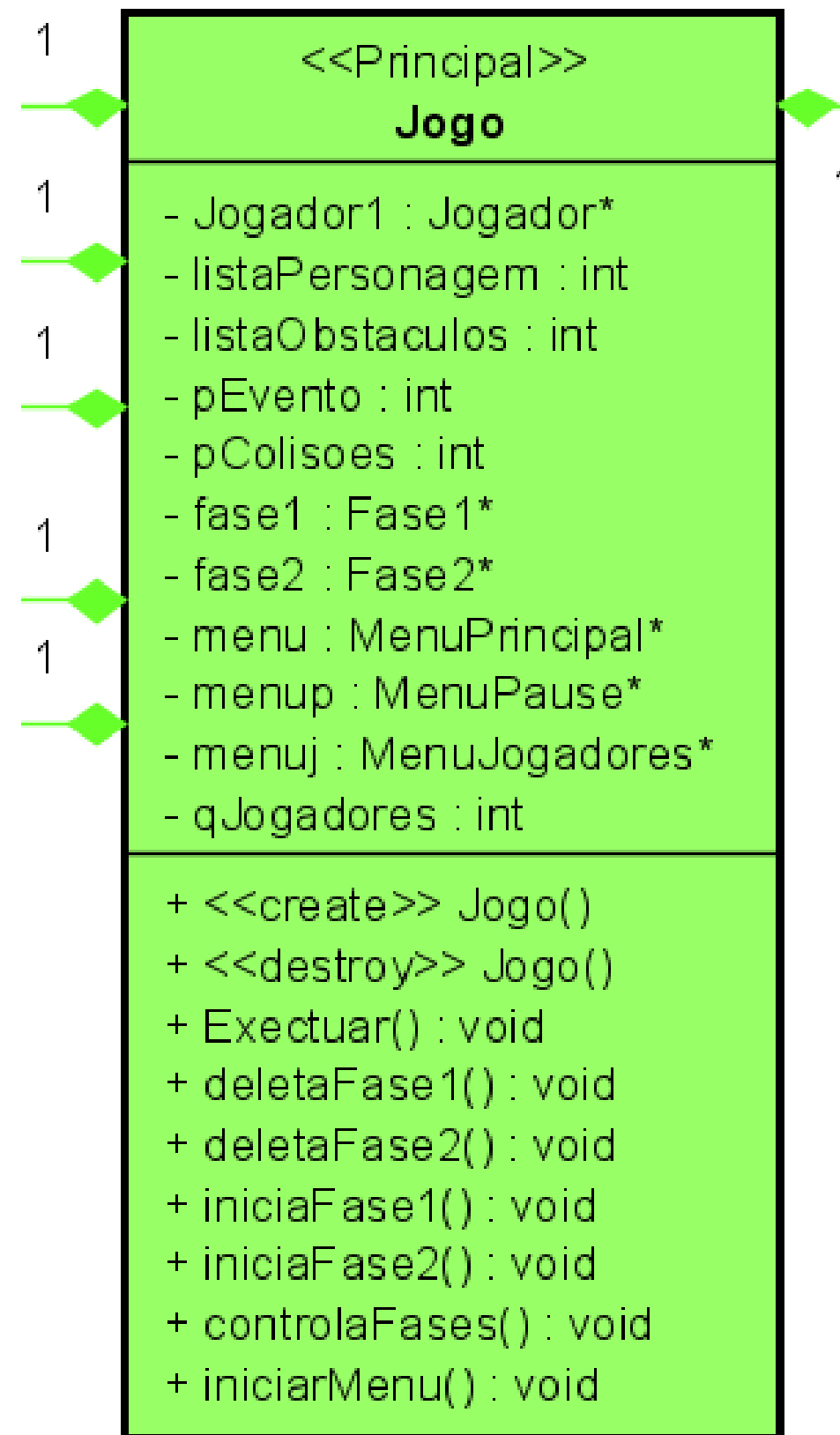























Tabela de Conceitos


N.	Conceitos	Status
1. 1	<ul style="list-style-type: none">- Classes, objetos. &- Atributos (privados), variáveis e constantes. &- Métodos (com e sem retorno).	
1. 2	<ul style="list-style-type: none">- Métodos (com retorno <i>const</i> e parâmetro <i>const</i>). &- Construtores (sem/com parâmetros) e destrutores	
1. 3	<ul style="list-style-type: none">- Classe Principal.	
1. 4	<ul style="list-style-type: none">- Divisão em .h e .cpp.	

2.1	<ul style="list-style-type: none"> - Associação direcional. & - Associação bidirecional. 	
2.2	<ul style="list-style-type: none"> - Agregação via associação. & - Agregação propriamente dita. 	
2.3	<ul style="list-style-type: none"> - Herança elementar. & - Herança em diversos níveis 	
2.4	<ul style="list-style-type: none"> - Herança múltipla. 	
3.1	<ul style="list-style-type: none"> - Operador this para fins de relacionamento bidirecional. 	
3.2	<ul style="list-style-type: none"> - Alocação de memória (new & delete) 	
3.3	<ul style="list-style-type: none"> -Gabaritos/Templates criada/adaptados pelos autores (e.g., Listas Encadeadas via Templates). 	
3.4	<ul style="list-style-type: none"> - Uso de Tratamento de Exceções (try catch). 	
4.1	<ul style="list-style-type: none"> - Construtoras e Métodos. 	

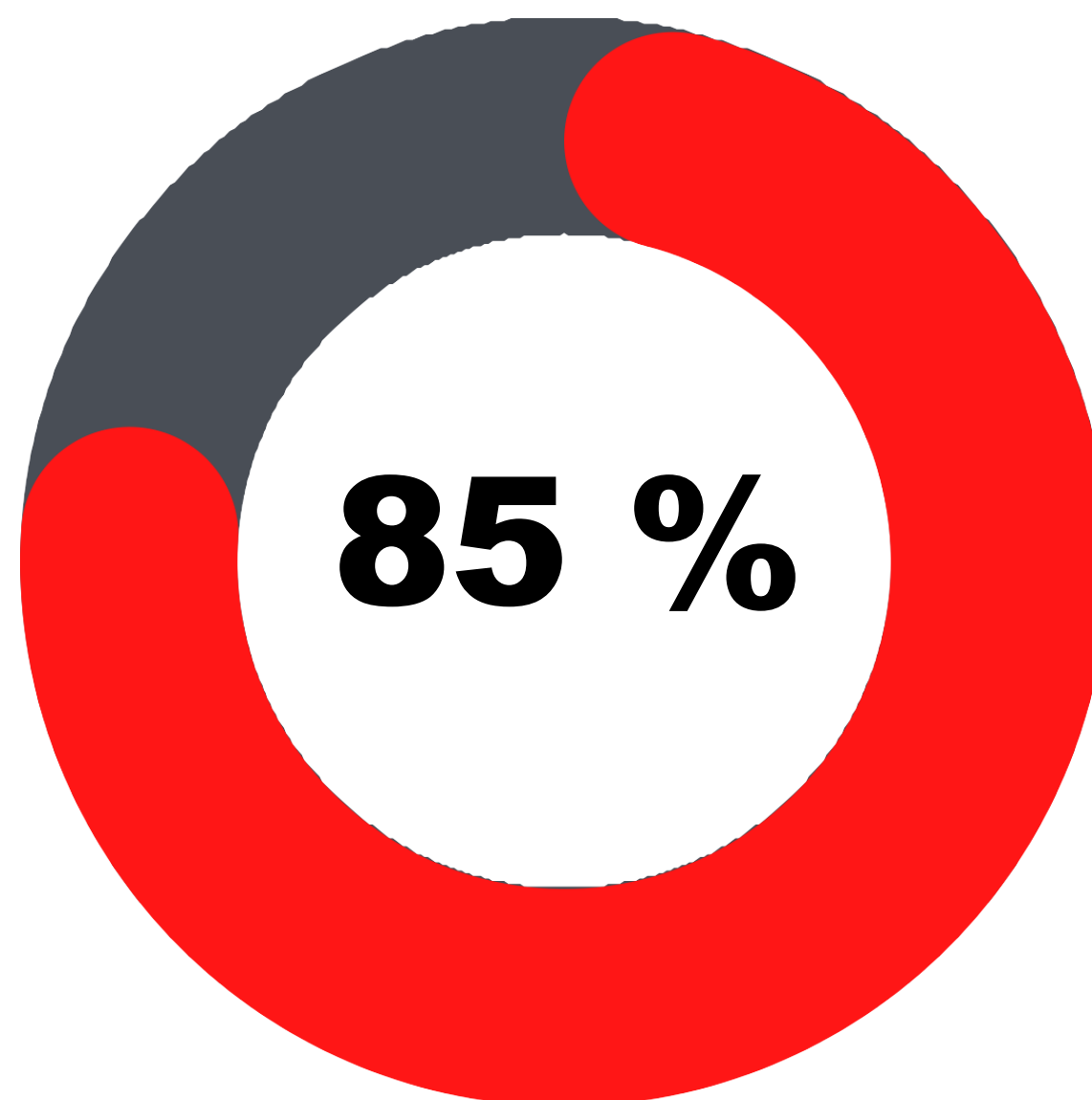
4.2	- Operadores (2 tipos de operadores pelo menos – Quais?).	
4.3	- Persistência de Objetos.	
4.4	- Persistência de Relacionamento de Objetos.	
5.1	- Métodos Virtuais Usuais.	
5.2	- Polimorfismo.	
5.3	- Métodos Virtuais Puros / Classes Abstratas.	
5.4	Coesão/Desacoplamento efetiva e intensa com o apoio de padrões de projeto.	
6.1	- Espaço de Nomes (Namespace) criada pelos autores.	
6.2	- Classes aninhadas (Nested) criada pelos autores.	

6.3	- Atributos estáticos e métodos estáticos.	
6.4	- Uso extensivo de constante (const) parâmetro, retorno, método...	
7.1	- A classe Pré-definida String ou equivalente. & - Vector e/ou List da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores)	
7.2	- Pilha, Fila, Bifila, Fila de Prioridade, Conjunto, Multi-Conjunto, Mapa OU Multi-Mapa.	
7.3	- <i>Threads</i> (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando Posix, C-Run-Time OU Win32API ou afins	
7.4	- <i>Threads</i> (Linhas de Execução) no âmbito da Orientação a Objetos com uso de Mutex, Semáforos, OU Troca de mensagens.	
8.1	- <i>Programação orientada e evento efetiva (com gerenciador apropriado de eventos inclusive) em algum ambiente gráfico.</i> OU - RAD – Rapid Application Development (Objetos gráficos como formulários, botões etc).	
8.2	- <i>Funcionalidades Elementares. & Funcionalidades Avançadas como: tratamento de colisões , duplo buffer</i>	

8.3	Ensino Médio Efetivamente.	
8.4	- Ensino Superior Efetivamente.	
9.1	- Compreensão, melhoria e rastreabilidade de cumprimento de requisitos. &	
9.2	- Diagrama de Classes em <i>UML</i> .	
9.3	- Uso efetivo e intensivo de padrões de projeto <i>GOF</i> , <i>i.e.</i> , mais de 5 padrões.	
9.4	- Testes à luz da Tabela de Requisitos e do Diagrama de Classes.	
10.1	- Controle de versão de modelos e códigos automatizado (via github e/ou afins). & - Uso de alguma forma de cópia de segurança (<i>i.e.</i> , <i>backup</i>).	
10.2	- Reuniões com o professor para acompanhamento do andamento do projeto.	
10.3	- Reuniões com monitor da disciplina para acompanhamento do andamento do projeto	

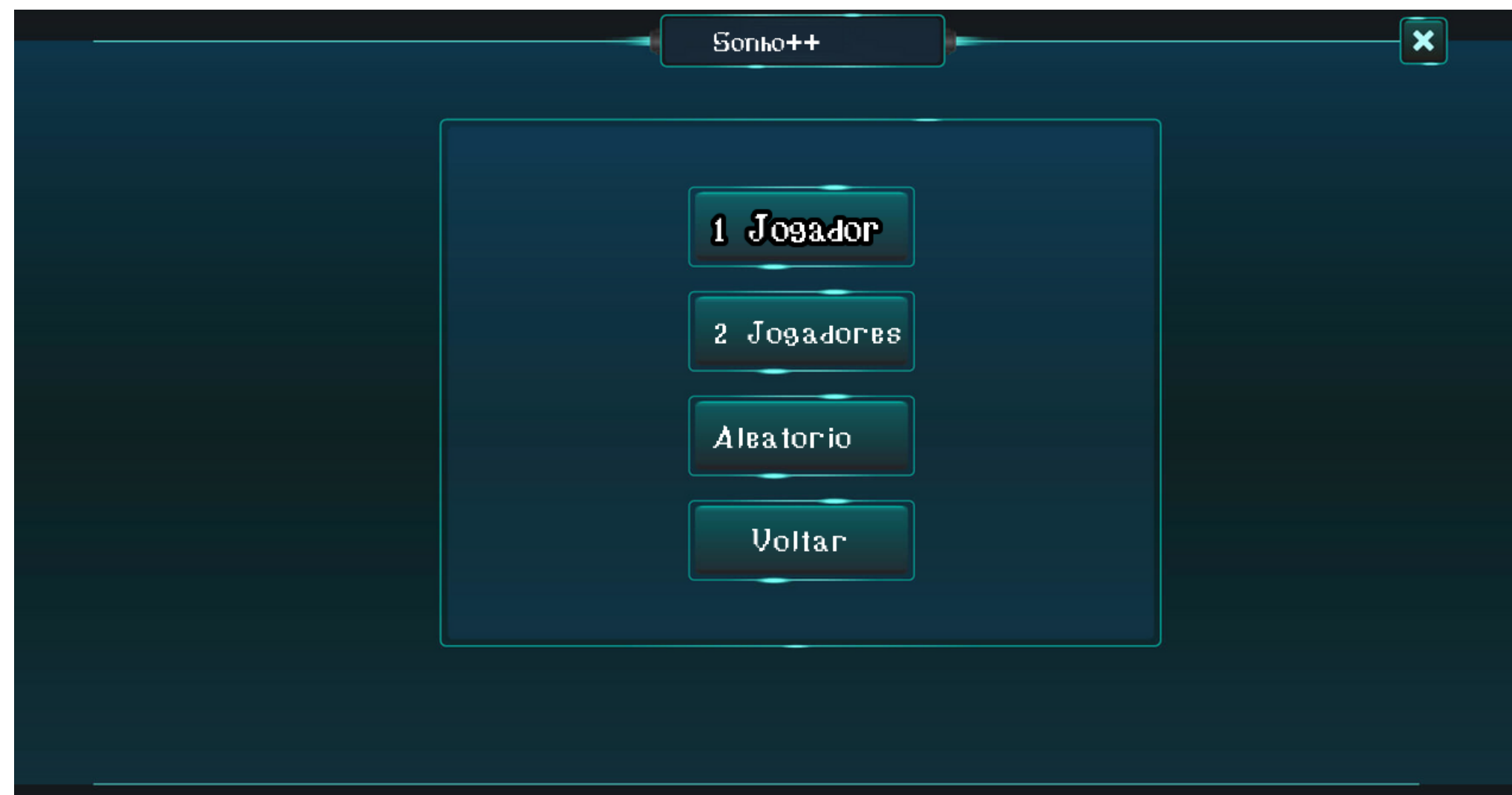
10.4	Revisão do trabalho escrito de outra equipe e vice-versa.	
------	---	---

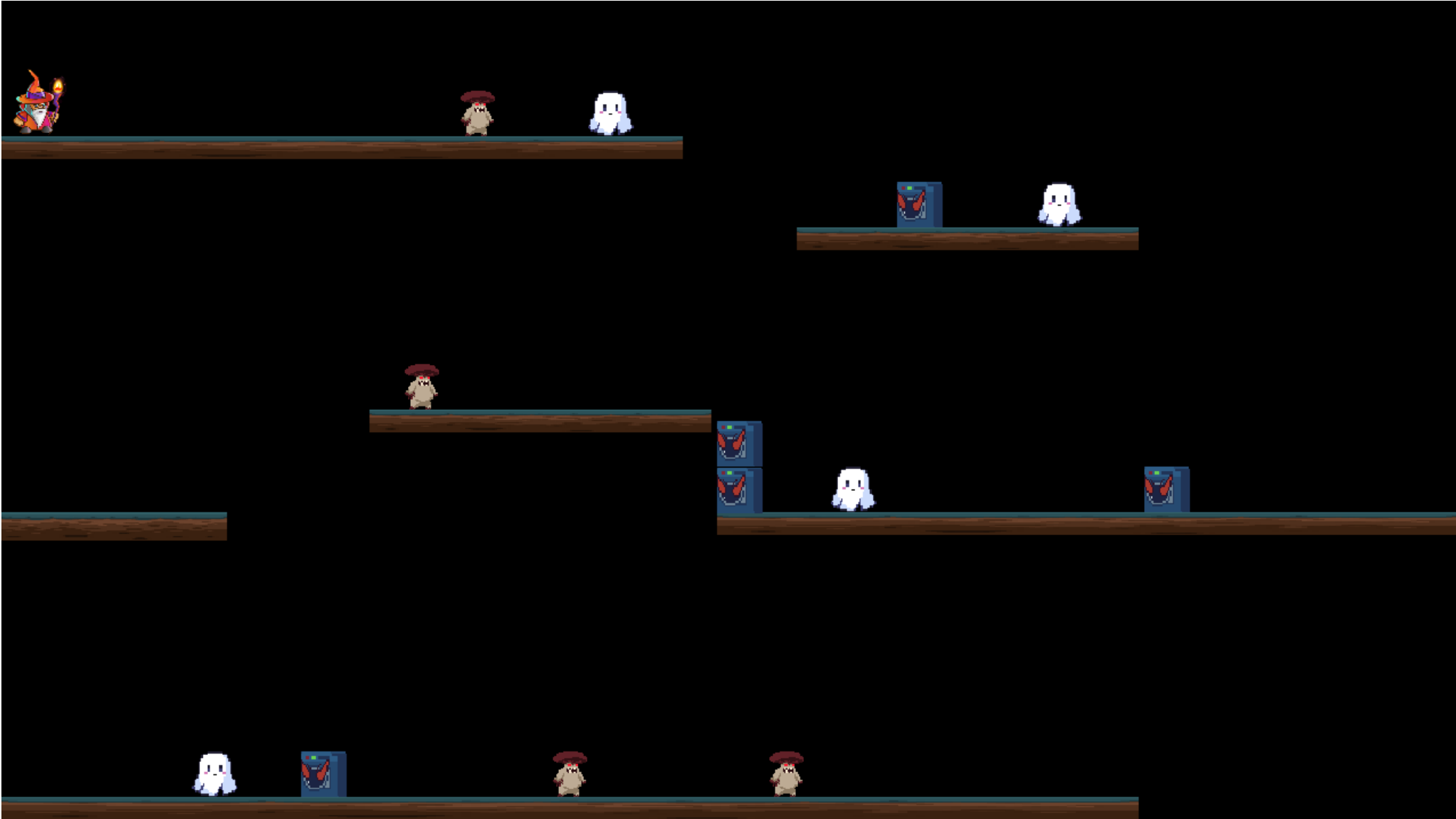
Percentual de Conceitos cumpridos

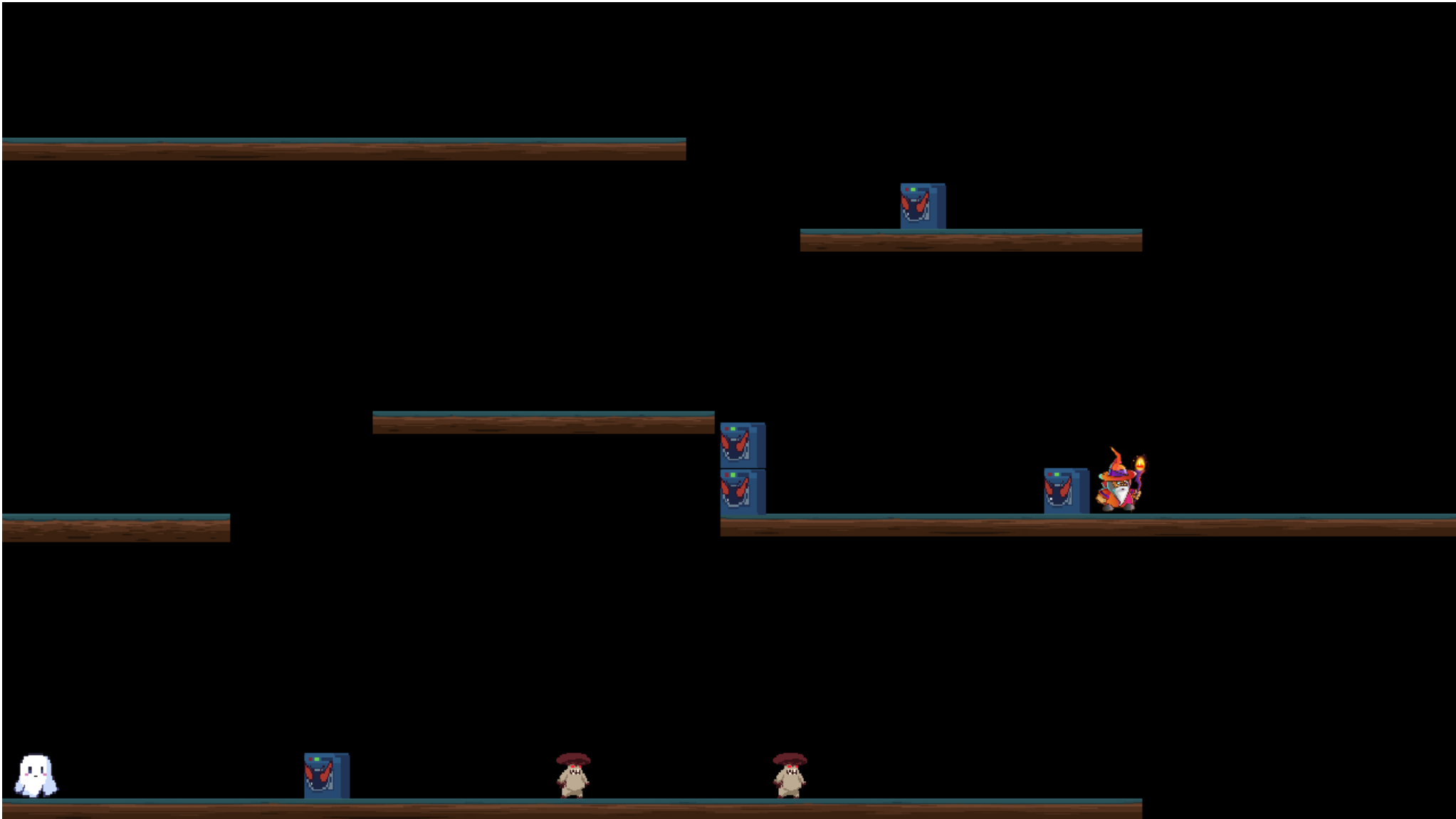


Jogo Executando







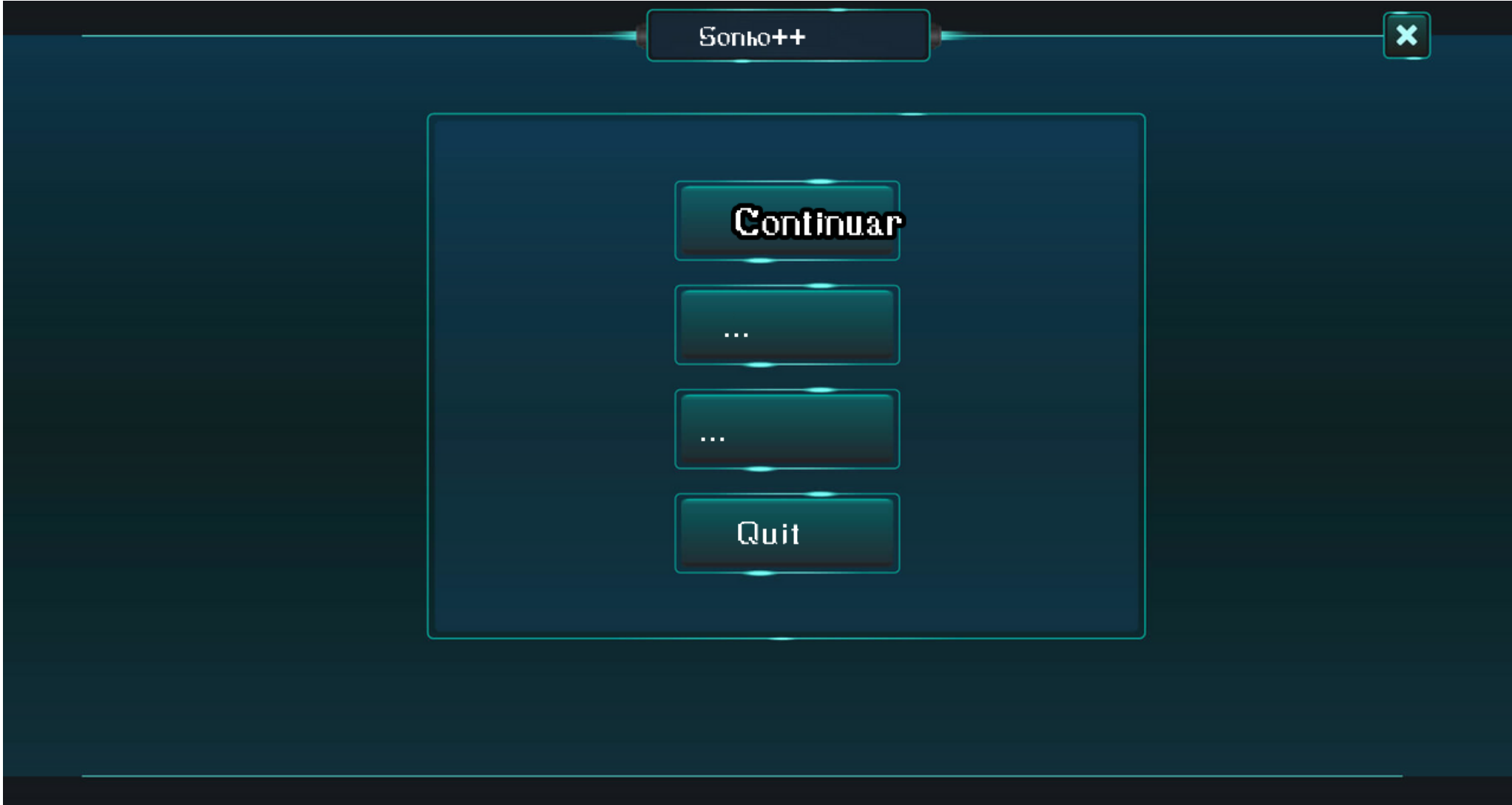












Conclusão Geral acerca dos Resultados

Apesar do tempo exíguo, tem-se em mãos um jogo com menus, personagens texturizados, gerenciamento de colisões, gerenciamento de projéteis e mais tarefas relativas a um jogo básico de plataforma. Assim sendo, os membros da equipe consideram o resultado satisfatório.