UAlg ISE
UNIVERSIDADE DO ALGARVE
INSTITUTO SUPERIOR DE ENGENHARIA

Tecnologias de Mercado

Trabalho final

Introdução

O objetivo deste trabalho final é implementar um aplicação local que simula os eventos de um jogo de futebol.

Será necessário complementar código previamente fornecido com classes que representem as entidades envolvidas numa partida.

Neste trabalho, é esperado que sejam aplicados todos os conhecimentos adquiridos ao longo da disciplina.

Anexo a este enunciado são fornecidos dois packages Java, um com classes para a *User Interface,* e outro com a classe onde será implementada toda a lógica de carregamento de eventos (a abordar mais à frente neste documento).

Contexto

No contexto deste trabalho, considere-se uma partida de futebol. Uma partida é composta por:

- Duas equipas, uma equipa da casa e uma equipa visitante (*home team* e *away team* respetivamente. Uma equipa é definida por um nome e um número fixo de jogadores;
- Um conjunto de eventos. Um evento corresponde à atividade que pode ocorrer num determinado momento de uma partida (Posse de bola, golo, lançamento lateral, substituições, etc.);
- Resultado do jogo;

Eventos numa partida(1/2)

Como já referido um evento refere-se a uma atividade que pode acontecer num determinado momento de uma partida. Considere que por cada um dos 90 minutos de uma partida há 90% de hipóteses de acontecer um evento. Esses eventos podem ser:

- Posse de bola – Apenas definido por uma descrição que esclarece que um determinado jogador de uma determinada equipa tem posse (probabilidade de acontecer: 60%);
- Golo – é incrementado o resultado para a equipa que marcou o golo (15% de probabilidade de acontecer);
- Substituição – um jogador de uma equipa é substituído por outro retirado da interface de ficheiro (introduzida mais à frente), se não houver mais jogadores este é apenas removido da equipa (10% de probabilidade de acontecer)

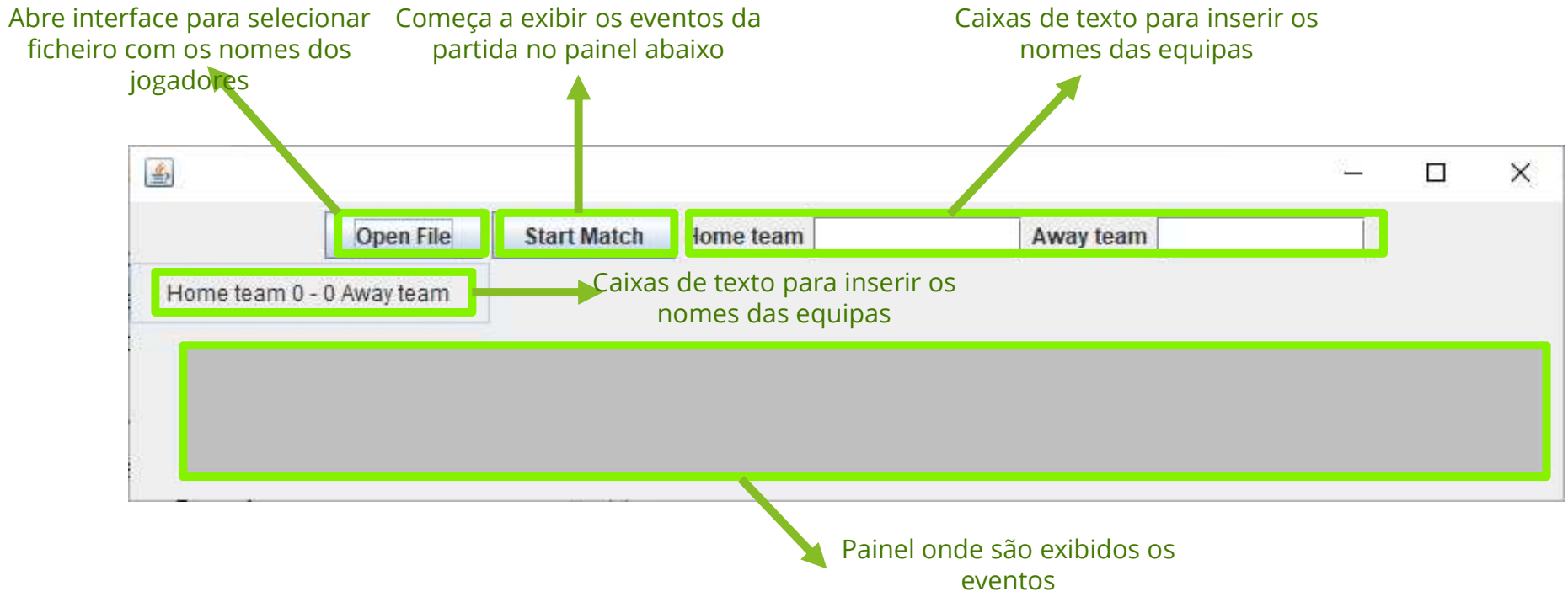
Eventos numa partida(2/2)

- Grande penalidade – Definida por uma descrição que refere que foi provocada uma grande penalidade por um jogador, e a probabilidade de ser golo do próximo evento é de 50% com os outros 50% de serem da grande penalidade ser falhada (5% de probabilidade de acontecer);

Interface Gráfica

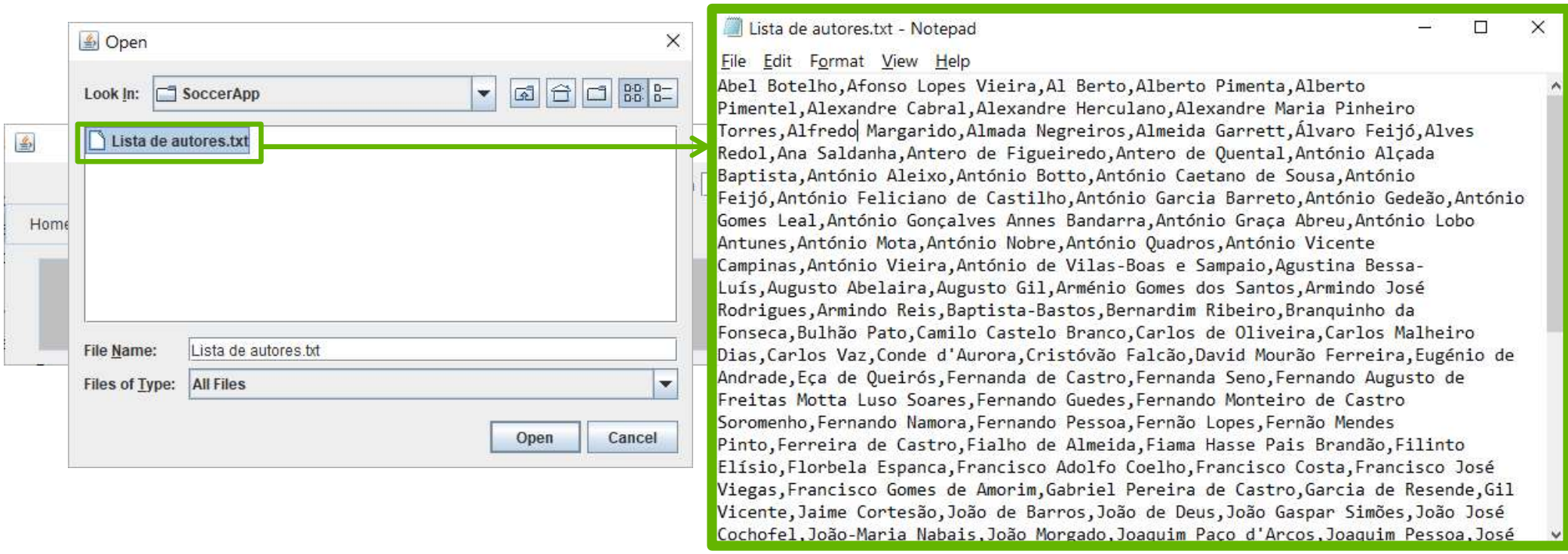
Com este enunciado é fornecida uma aplicação gráfica já implementada em Java swing. Não é espectável que esta interface seja modificada.

A janela apresentada é composta pelos seguintes elementos:



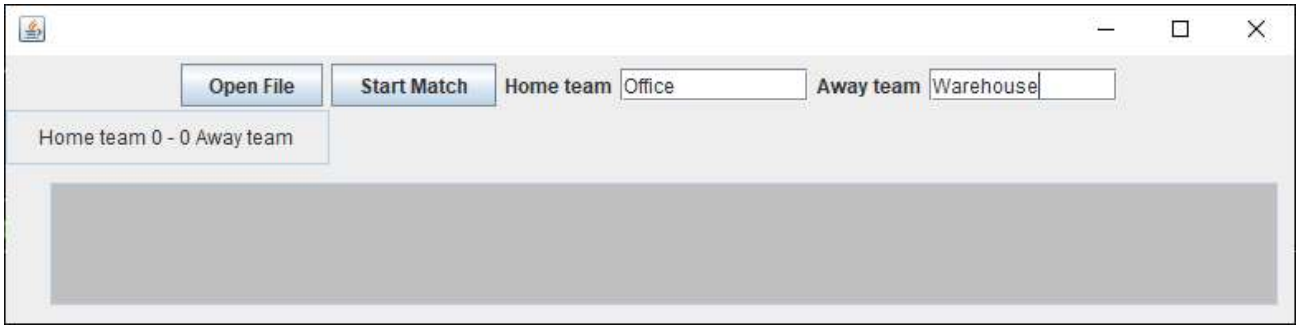
Funcionamento da aplicação

Ao executar a aplicação, é apresentada a janela vista anteriormente. O primeiro passo é importar um ficheiro com os nomes dos jogadores. Este ficheiro, deve ser um ficheiro de texto simples, com vários nomes separados por virgulas.



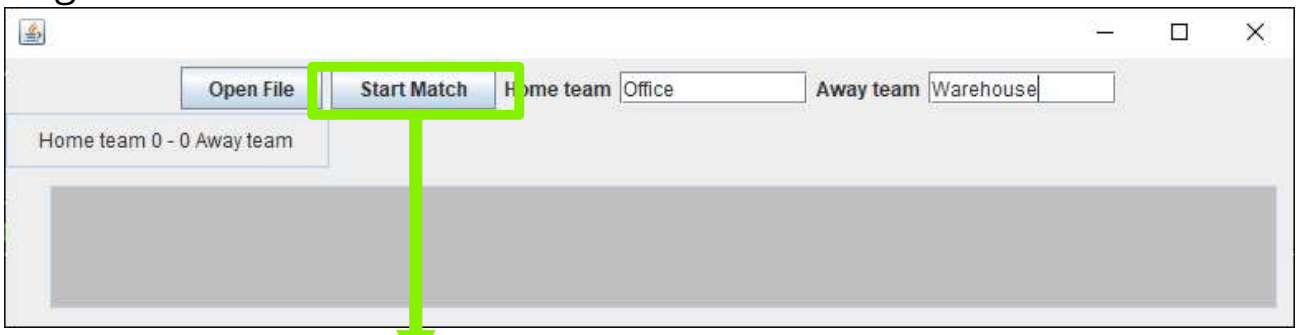
Funcionamento da aplicação(1/2)

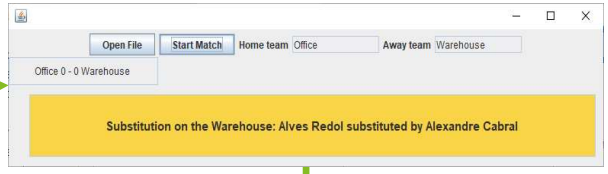
Ao abrir o ficheiro, é necessário inserir os nomes das equipas, nas caixas de texto criadas para o efeito:



Funcionamento da aplicação(2/2)

Ao clicar no botão para começar a partida, os eventos são apresentados imediatamente com o intervalo de um segundo:





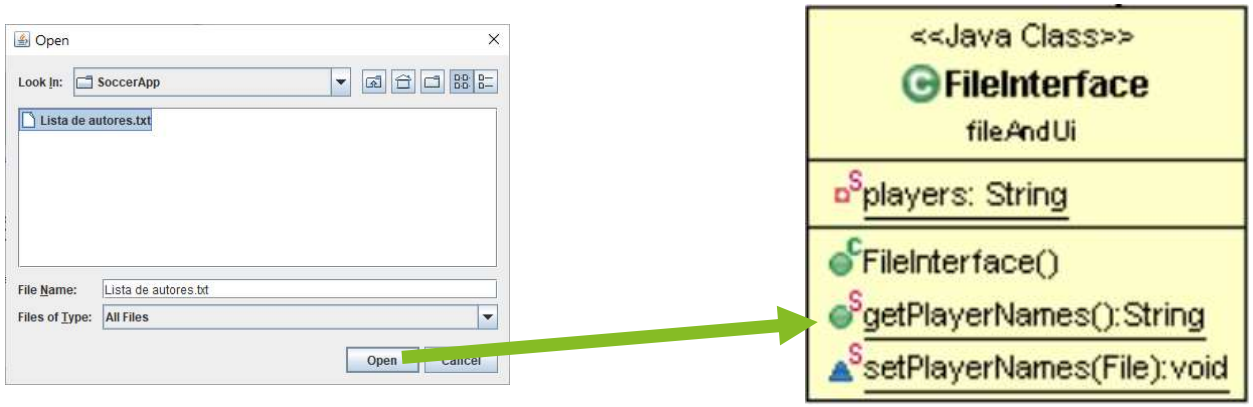
(. . .)

Deep dive

No material fornecido, pode ser encontrado o diagrama de classes do software implementado até à altura. No entanto este enunciado vai-se focar em três classes:

- FileInterface – Onde vai ser disponibilizado os nomes dos jogadores;
- IGameEvent – Um contrato definido para a aplicação gráfica, que representa um evento a ser apresentado na UI;
- EventLoader – onde deve ser implementada toda a lógica referente ao que deve ser retornado;

Deep dive - FileInterface



Ao abrir o ficheiro, assim que se clica no botão "*Open*" o método estático *getPlayerNames* retorna o conteúdo integral do ficheiro. Note que não é feita qualquer validação de integridade ao conteúdo do ficheiro, o texto que está nele contido é disponibilizado nesta classe estática.

Deep dive – IGameEvent

A interface *IGameEvent*, fornece os detalhes da partida a cada instante à aplicação gráfica. Cada um dos métodos de uma instância que seja também do tipo *IGameEvent* fornece os dados ilustrados abaixo:

Deep dive – EventLoader

Uma instância de EventLoader, representa o conjunto de eventos de uma partida individual. Por cada vez que clicamos no botão *Start Game* é criada sempre uma nova instância de EventLoader, e devem ser gerados novamente os eventos.

É nos métodos de EventLoader que deve ser implementado o código para criar os eventos da interface gráfica, com base na API implementada (isto é, as classes definidas por si para representar uma partida). Por defeito, as chamadas aos métodos de EventLoader lançam a exceção *UnsupportedOperationException* pois estes métodos estão apenas declarados e não implementados.

- *setUpEvents* – neste método deve ser definido o conjunto de eventos encapsulados pelo tipo *EventLoader.* Nos parâmetros, é recebido os nomes das equipas e o número de jogadores por equipa. A lógica que passar os nomes das equipas, recebidos na UI, por parâmetro já está implementada e o numero de jogadores por equipa está definido na constante *NUMBER_OF_PLAYERS* definida na classe SoccerFrame. É garantido que tem estes campos disponíveis quando estiver a implementar o seu código.
- getGameEvents – retorna um *array* com os eventos definidos acima;

EventLoader.java

```
1  package soccerapi.utility;
2
3  import fileAndUi.IGameEvent;
4
5  public class EventLoader {
6
7
8      public void setUpEvents(String homeTeam,String awayTeam,int numberOfPlayers) {
9
10         //*****************************************************//
11         //*****************IMPLEMENTAR AQUI*****************//
12         //*****************************************************//
13
14         throw new UnsupportedOperationException();
15     }
16
17     public IGameEvent[] getGameEvents() {
18         //*****************************************************//
19         //*****************IMPLEMENTAR AQUI*****************//
20         //*****************************************************//
21         throw new UnsupportedOperationException();
22     }
23 }
24
```

Notas finais

- Durante a implementação tenha em conta, que há parâmetros que para efeitos de discussão podem ser alterados;
- Defina os seus tipos, tento em conta que a sua *API* pode ser usada noutras aplicações que não nesta UI;
- O relatório deve contar a descrição de todas as decisões tomadas ao longo do trabalho, deve conter um diagrama de classes UML e diagramas de objectos UML devem ser desenhados para expor o estado da aplicação quando necessário;

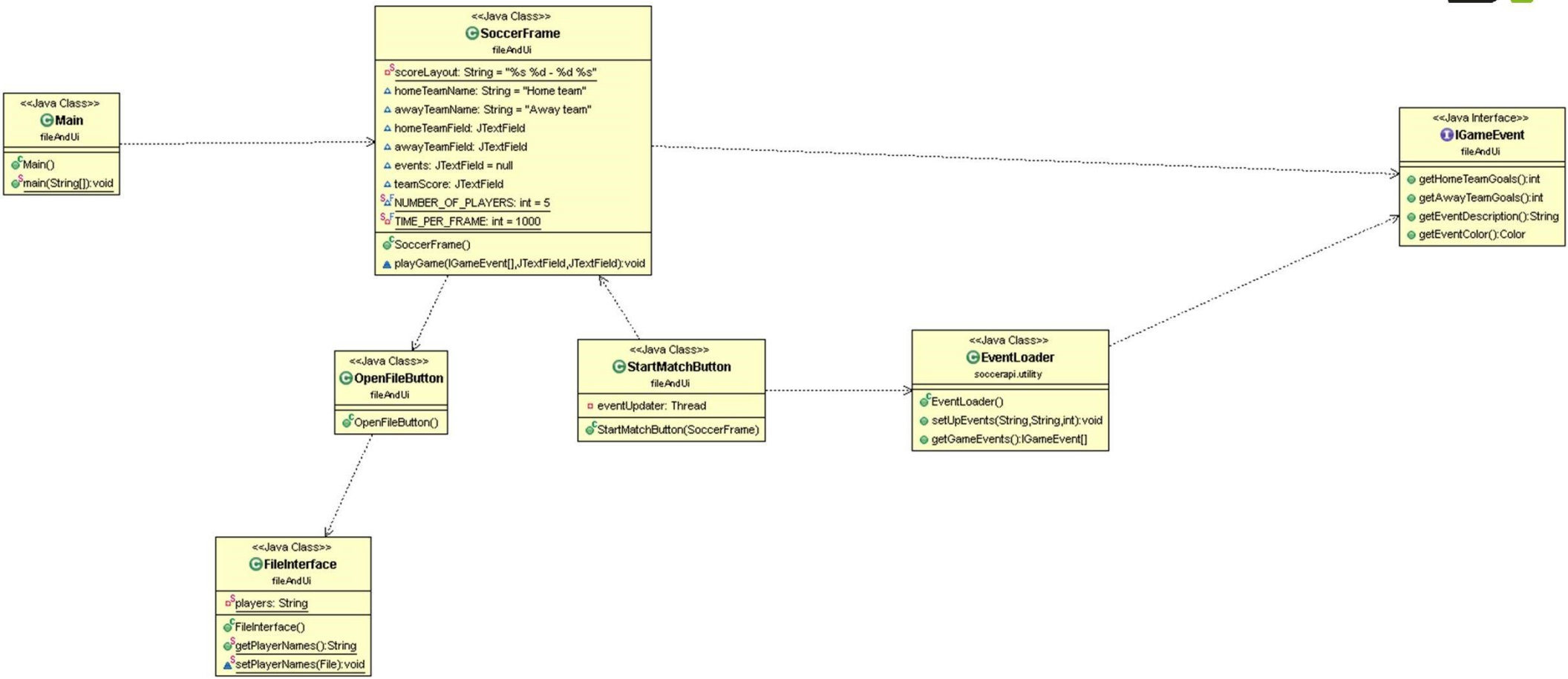
Bom trabalho!

Data limite de entrega (relatório e código)

20 de Fevereiro

Anexos

Diagrama de classes UML



Probabilidades dos eventos

Evento	Probabilidade
Posse de bola	60%
Golo	15%
Substituição	10%
Grande Penalidade	5% (com 50% de probabilidade de ser Golo)

