Projecto Guiado –5ª Iteração
Documentação com Javadoc e UML

5.1 Documentar código fonte do projecto com Javadoc

Garantir que cada classe é precedida de comentários de documentação no formato Javadoc (/** … */) explicando as responsabilidades principais da classe.

De acordo com o tempo disponível, documentar os métodos mais relevantes com comentários em Javadoc.

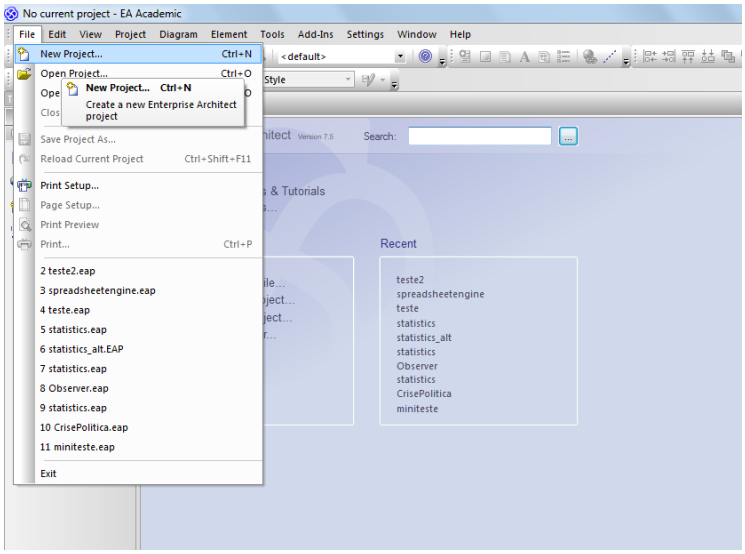
Referências:

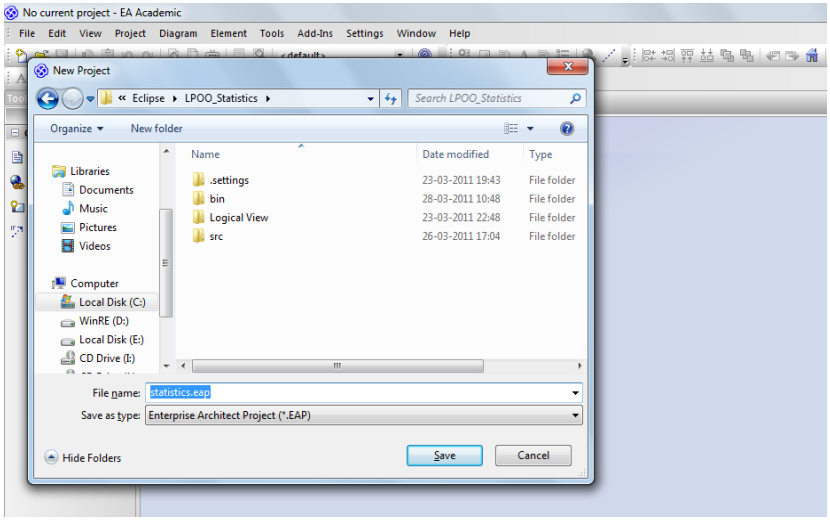
- http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html
- http://www.devdaily.com/java/edu/pj/pj010014

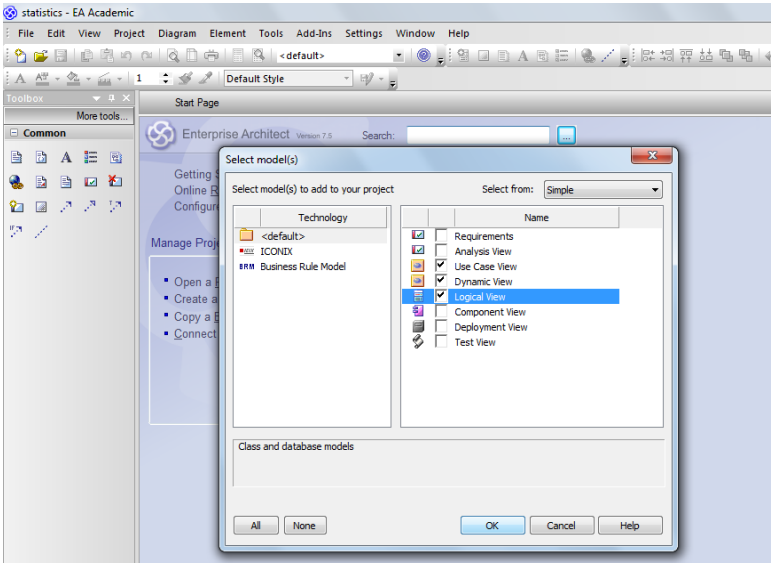
5.2 Documentar estrutura do projecto com UML (diag. classes)

a) Criar um novo projecto no Enterprise Architect com *logical view* (e opcionalmente outras vistas).

Exemplo:

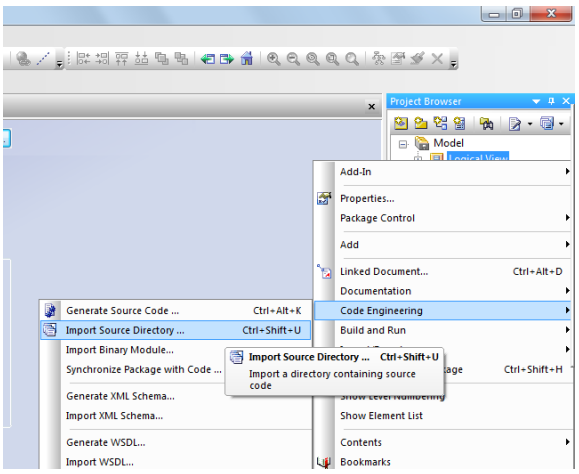


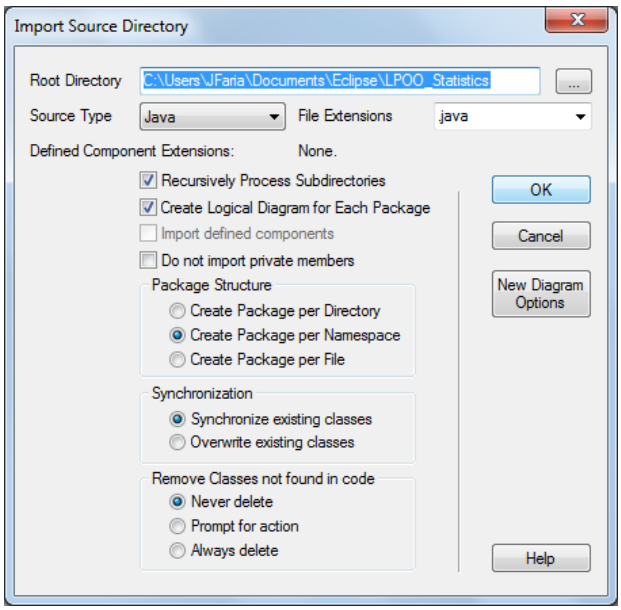




b) Importar para a vista lógica o código Java do projecto.

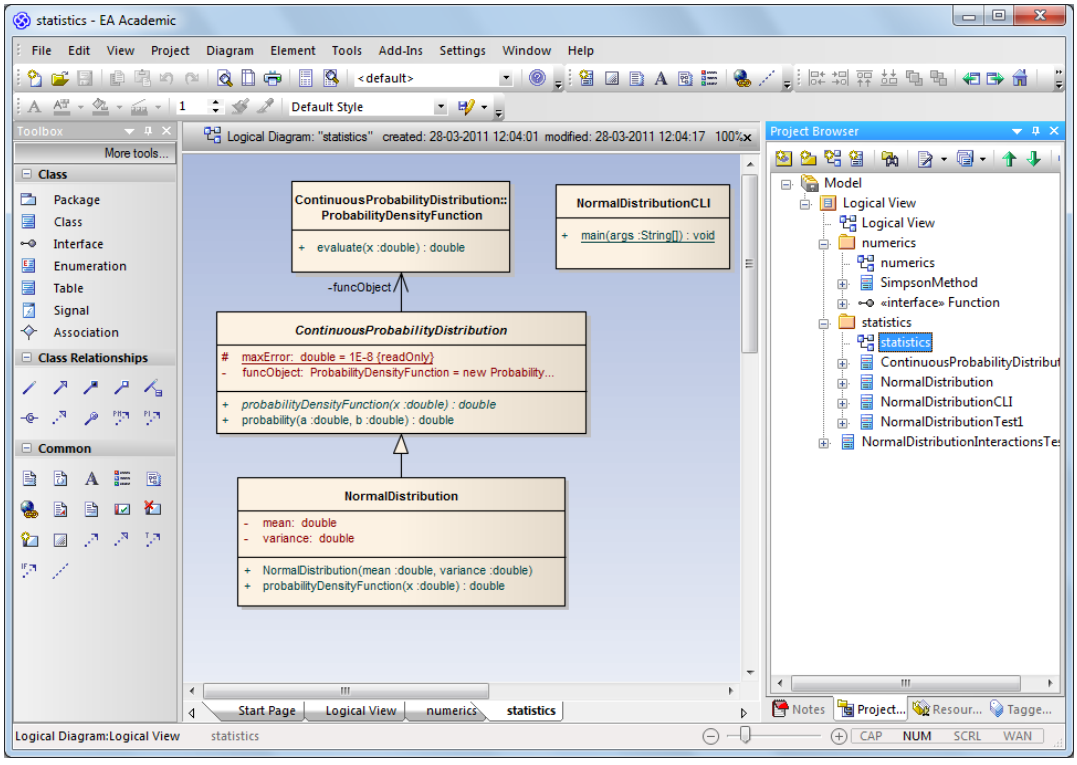
Exemplo:





c) Visualizar o modelo obtido.

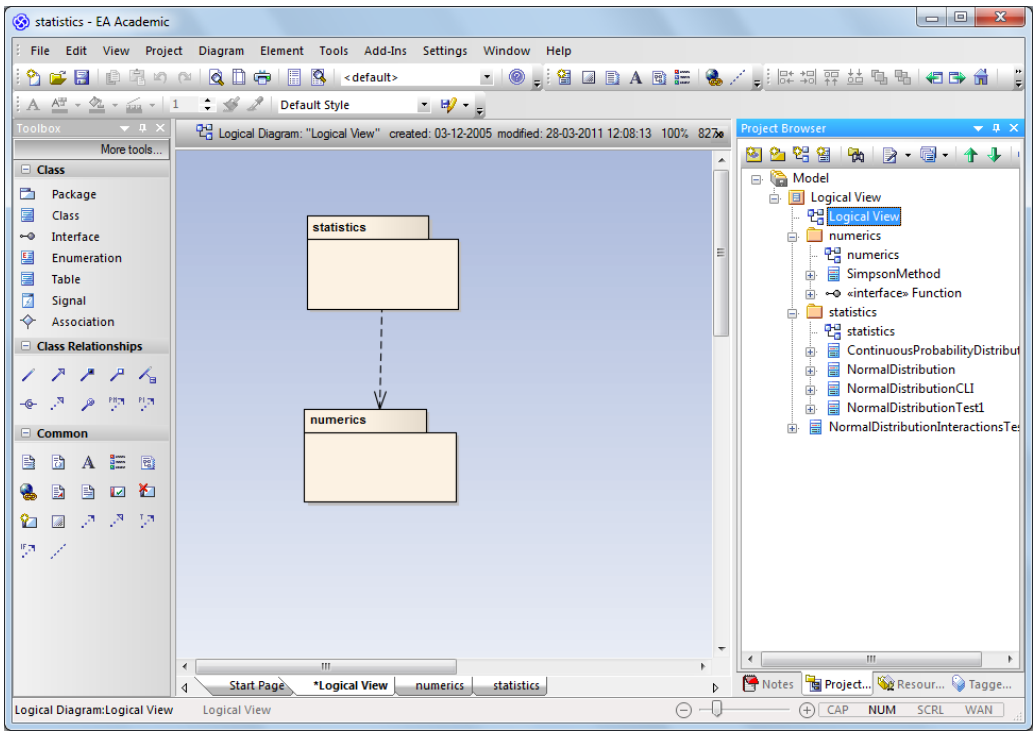
Exemplo:



Nota: os comentários Javadoc são também importados para notas.

d) Criar diagrama de classes de alto nível, apenas com *packages* e dependências entre packages.

Exemplo:



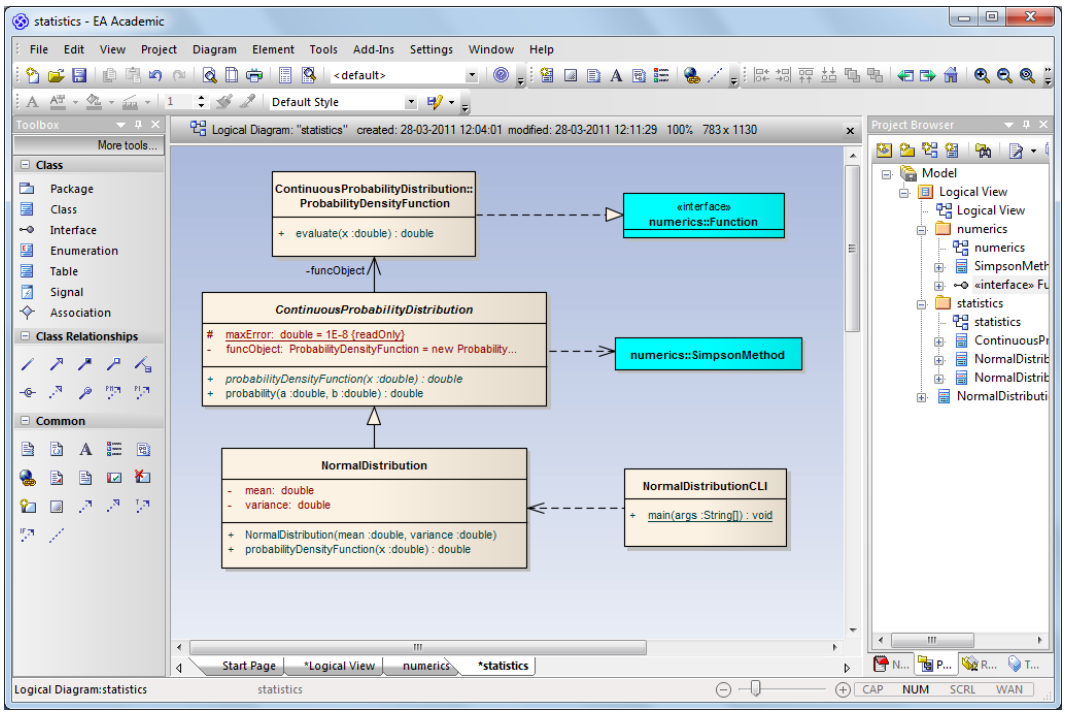
e) Completar diagrama de classes dentro de cada *package*.

Acrescentar dependências entre classes não descobertas pelo EA.

Acrescentar ligações com classes doutros *packages* (escondendo as features dessas classes).

Corrigir associações entre classes (o EA pode não descobrir asscições e multiplicidade corretamente).

Exemplo:



5.3 Documentar comportamento com UML

No modelo UML, debaixo da classe que representa o herói, criar um diagrama de estados (State Machine) modelando o seu ciclo de vida, seguindo os seguintes passos:

i. Começar por identificar os estados possíveis (exemplo: Sem espada, Com espada, Matou todos os dragões, Morto por um dragão, Encontrou saída), bem como o estado inicial.

ii. Identificar depois as transições possíveis e os eventos que causam essas transições (exemplo: Encontrou a espada, Encontrou um dragão, Encontrou a saída, Um dragão acorda). No Enterprise Architect, o evento é indicado no campo *trigger* do tabulador *Constraints.* Sempre que num estado não é indicada a resposta a um evento, assume-se que ou o evento não pode acontecer nesse estado ou é ignorado.

iii. No caso de o mesmo evento poder conduzir a diferentes estados, ou no caso de além do evento ser necessária alguma condição para a transição ocorrer, é necessário indicar condições de guarda (exemplos: quando encontra um dragão e está armado, mata o dragão e o estado seguinte depende de existirem ou não mais dragões; quando encontra um dragão e está desarmado, o estado seguinte depende do dragão estar ou não a dormir; quando o dragão acorda e o herói está desarmado, o efeito depende de o dragão estar ou não junto ao herói). No Enterprise Architect, a condição de guarda é indicada no campo *guard* do tabulador *Constraints.*

iv. Indicar também ações do herói associadas às transições (exemplos: apanhar a espada, matar o dragão, ser morto pelo dragão, sair do labirinto). No Enterprise Architect, a condição de guarda é indicada no campo *effect* do tabulador *Constraints.*

v. Uma vez que os eventos acima sugeridos são de relativo alto nível, é conveniente clarificar esses eventos em função de eventos de mais baixo nível (exemplo: Encontra dragão = herói movimenta-se para uma posição adjacente a um dragão, ou um dragão movimenta-se para uma posição adjacente ao herói), o que poder ser efetuado em notas do diagrama.

5.4 Entregar projecto

Submeter pelo moodle ou email (zip) para o docente da aula prática os ficheiros finais do projecto até ao fim do dia 15 de Abril:

- código fonte .java;
- modelo UML .eap;
- outros ficheiros necessários.

Antes de enviar, garantir que:

- Todas as funcionalidades pedidas estão implementadas (incluindo os testes em JUnit e interface por linha de comando);
- Todas as classes estão documentadas como referido acima;
- O modelo UML foi devidamente construído como referido acima.

O trabalho será avaliado com base nos seguintes parâmetros, por ordem:

1. Cumprimento das funcionalidades pedidas;
2. Estruturação do código;
3. Documentação (Javadoc, UML);
4. Qualidade da interação com o utilizador