

PROGRAMAÇÃO ORIENTADA PELOS OBJECTOS

Implemente as suas classes

2

- Escolha uma ordem que facilite o ensaio incremental da sua aplicação
- Evite deixar “pontas soltas”
- Lembre-se **sempre** da regra da versão estável
- Vá comentando o seu código à medida que o escreve
 - ▣ A experiência mostra que comentários deixados para o fim acabam por não ser feitos, ou são feitos à pressa
 - ▣ Comentar o cabeçalho dos métodos antes de os desenvolver obriga-o a pensar no que tem de fazer, antes de os começar a programar

Implementação

3

```

1 package poo;
2
3 import java.util.Scanner;
4
5 /**
6  * Programa principal para demonstração da aplicação TheStupidFriendsBook.
7  * @author Miguel Goulão
8  */
9
10 public class Main {
11     // Comandos do utilizador
12     private static final String QUIT = "Sair";
13     private static final String NEW = "Novo";
14     private static final String ADD_FRIEND = "Adiciona";
15     private static final String REM_FRIEND = "Remove";
16     private static final String DO = "Faz";
17     private static final String VOTE = "Vota";
18     private static final String GOOD = "Bem";
19     private static final String BAD = "Mal";
20     private static final String LIST = "Lista";
21     private static final String BORING = "Aborrecido";
22     private static final String INTELLIGENT = "Inteligente";
23     private static final String STUPID = "Estúpido";
24     private static final String BANDIT = "Bandido";
25     private static final String ANGEL = "Anjinho";
26
27     // Feedback dado pelo programa
28     private static final String PROMPT = "> ";
29     private static final String OK = "Ok";

```

poo.Main

Programa principal para demonstração da aplicação TheStupidFriendsBook.

Author:
Miguel Goulão

Documentação

The screenshot shows the Eclipse IDE with the 'Person.java' file open. The code defines two methods: `addAction` and `vote`. The `vote` method is highlighted, and its documentation is displayed in the 'Declaration' view at the bottom.

```

17  boolean addAction(String description);
18
19  /**
20   * Vota na acção, incrementando os votos nos benefícios próprios
21   * se <code>goodForPerson</code> for <code>true</code>, ou decrementando-os,
22   * caso contrário, e fazendo o mesmo em relação aos benefícios alheios,
23   * com o argumento <code>goodForOthers</code>.
24   * @param description - descrição da acção a votar
25   * @param goodForPerson - <code>true</code>, se for benéfico para o próprio, <code>false</code> caso contrário
26   * @param goodForOthers - <code>true</code>, se for benéfico para os outros, <code>false</code> caso contrário
27   * @return <code>true</code>, se for possível votar, <code>false</code> caso contrário.
28   */
29  boolean vote(String description, boolean goodForPerson, boolean goodForOthers);
30
31  /**
32   * Devolve o nome da pessoa
33   * @return - o nome da pessoa.
34   */
35  String getName();
36
37  /**
38   * Devolve a personalidade dominante da pessoa.
39   * @return Devolve um inteiro representando o traço de personalidade dominante,
40   * com os valores <code>INTELLIGENT</code>, <code>STUPID</code>,
41   * <code>BANDIT</code>, ou <code>ANGEL</code>.
42   */

```

Declaration

• **boolean poo.Person.vote(String description, boolean goodForPerson, boolean goodForOthers)**

Vota na acção, incrementando os votos nos benefícios próprios se `goodForPerson` for `true`, ou decrementando-os, caso contrário, e fazendo o mesmo em relação aos benefícios alheios, com o argumento `goodForOthers`.

Parameters:

- description** - descrição da acção a votar
- goodForPerson** - `true`, se for benéfico para o próprio, `false` caso contrário
- goodForOthers** - `true`, se for benéfico para os outros, `false` caso contrário

Returns:

- `true`, se for possível votar, `false` caso contrário.

Convenções de documentação

5

- Escreva a documentação para as pessoas que vão ter de usar o seu código e mantê-lo
 - ▣ Documente cuidadosamente a interface pública do seu código para que outros o possam usar de forma correcta e eficiente
 - ▣ Documente cuidadosamente a interface privada e os detalhes de implementação internos, para que outros possam manter e melhorar o seu código

Convenções de documentação

6

- Assuma sempre que alguém completamente estranho ao seu código o vai ter de entender, mais cedo ou mais tarde!
 - ▣ Se passar tempo suficiente, até o autor de código mal documentado terá dificuldade em o compreender
- Mantenha o seu código e comentários sincronizados
 - ▣ Ambos fazem parte do produto, não menospreze nenhum dos dois
- Seja claro, conciso e preciso

Três tipos de comentário, em Java

7

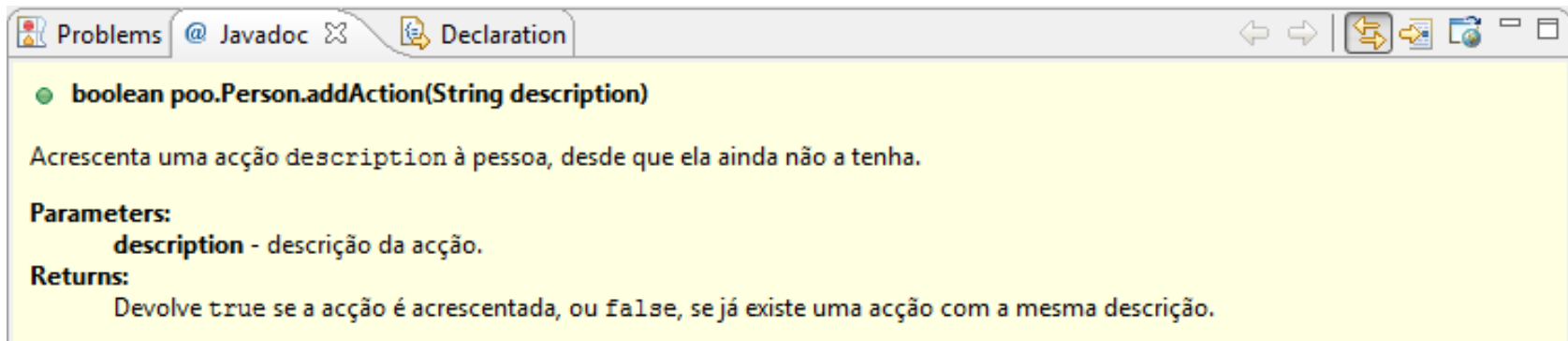
- Comentário de documentação
 - ▣ Começa em `/**` e termina em `*/`
- Comentário standard (estilo emprestado do C)
 - ▣ Começa em `/*` e termina em `*/`
- Comentário de uma linha
 - ▣ Começa em `//` e termina no fim da linha

Comentário de documentação

8

□ Exemplo:

```
/**  
 * Acrescenta uma acção <code>description</code> à pessoa, desde que ela  
 * ainda não a tenha.  
 * @param description - descrição da acção.  
 * @return Devolve <code>true</code> se a acção é acrescentada,  
 * ou <code>false</code>, se já existe uma acção com a mesma descrição.  
 */  
boolean addAction(String description);
```



Comentário de documentação

9

- Pode ser usado para comentar qualquer elemento
 - ▣ Declaração de interface, classe, método, construtor, campo
- Estes comentários dão a informação necessária para que o Javadoc possa gerar documentação em formato html (ao estilo que costuma ver na documentação da API do Java)
- O Javadoc apenas reconhece estes comentários quando eles aparecem imediatamente antes do elemento a comentar.

Comentário de documentação

10

- ❑ O Javadoc não reconhece estes comentários fora do sítio esperado
 - ▣ Não use este formato para comentar detalhes de implementação
- ❑ O Javadoc apenas reconhece um bloco de comentários por elemento
- ❑ O principal objectivo destes comentários é definir um contrato entre o cliente e o fornecedor de um serviço
 - ▣ Não coloque nestes comentários os detalhes de implementação

Comentários de documentação

11

- Escreva os comentários da interface de programação (API) **antes** de escrever o código
- Documente os membros *públicos, protegidos, de package e privados*, ou seja **todos**
- Escreva uma descrição sumária para cada package
 - ▣ É necessário criar um ficheiro de comentários de package especial denominado `package.html` e colocar esse ficheiro dentro do package, para que o Javadoc o encontre
- Escreva uma descrição sumária para cada aplicação, ou grupo de packages
 - ▣ É necessário criar um ficheiro de comentários `overview.html` e indicar ao Javadoc a sua localização

Estilo nos comentários de documentação

12

- Use um formato consistente ao longo de toda a documentação
 - ▣ Tipicamente, o comentário começa com um texto, auto-contido, descritivo para a entidade
 - ▣ Depois, usam-se várias tags Javadoc, seguidas do respectivo texto, para documentar detalhes como:
 - Parâmetros de operações
 - Valores a retornar pela operação
 - ...
- Envolver palavras chave, identificadores e constantes com as tags `<code> . . . </code>`.

Comentários standard (estilo C)

13

- Use este estilo de comentário **apenas para esconder temporariamente do compilador código**
 - ▣ Por exemplo, esta técnica é usada como passo intermédio, quando estamos a remover código e queremos ter a certeza de que é seguro fazer essa remoção **antes** de o apagar definitivamente

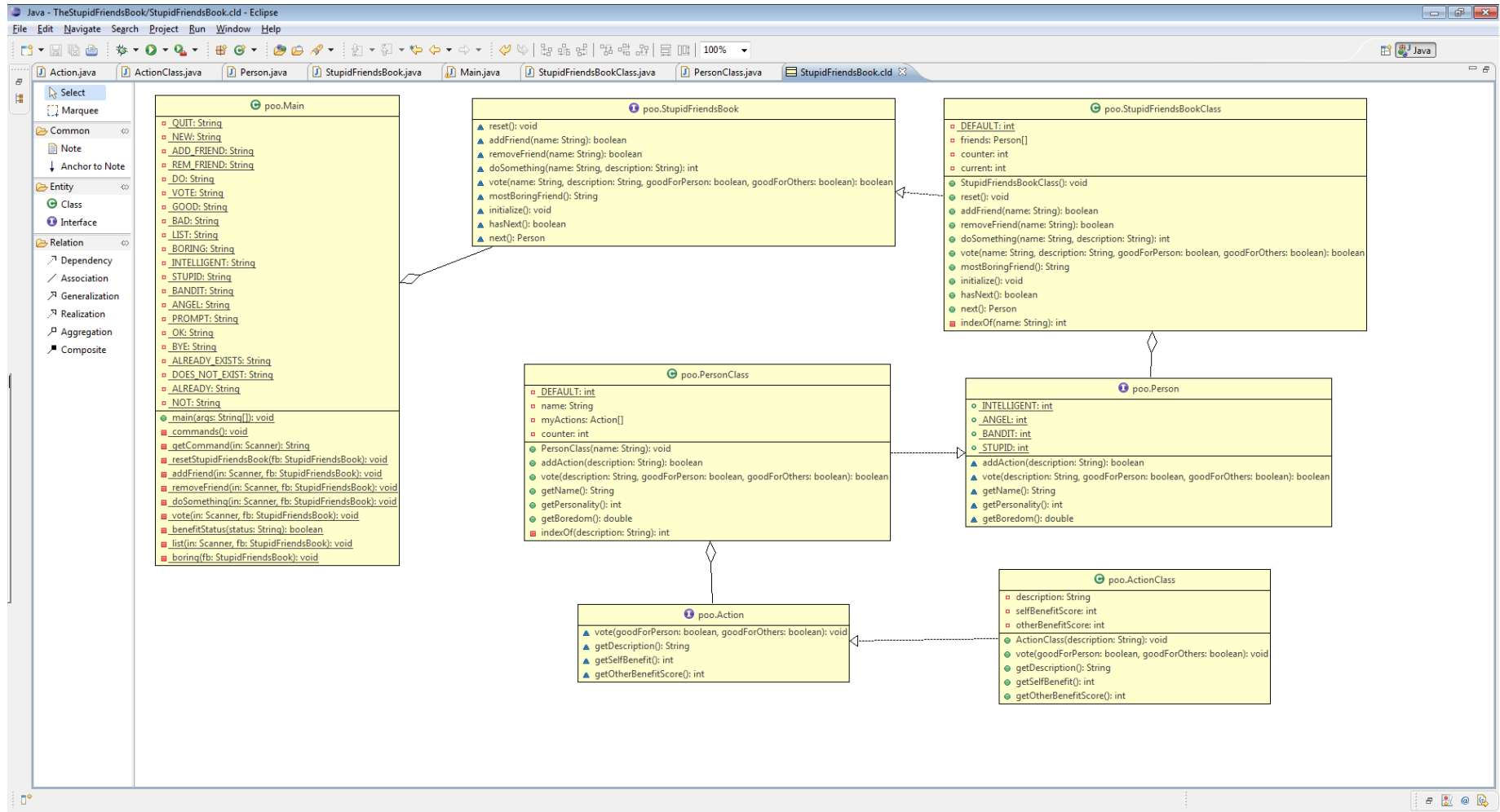
Comentário de uma linha

14

- Use este tipo de comentário para explicar detalhes de implementação
 - ▣ Use um ou mais comentários destes para documentar
 - O objectivo de variáveis ou expressões
 - Decisões ao nível da implementação
 - Informação adicional sobre algoritmos complicados
 - “Remendos” aplicados a defeitos
 - Código a necessitar de melhorias/optimização
 - Problemas conhecidos, limitações, ou deficiências
- Descreva **porque é que o código faz algo**, não o que o código faz
- Use comentários de fim de linha para documentar variáveis locais e chavetas de fecho (“}”), em estruturas de controle aninhadas complicadas

Diagrama de classes e interfaces

15



Entregáveis

16

- Código fonte do projecto
 - ▣ Package `poo`
 - `Main.java`
 - `StupidFriendsBook.java`
 - `StupidFriendsBookClass.java`
 - `Person.java`
 - `PersonClass.java`
 - `Action.java`
 - `ActionClass.java`
- Documentação em formato Javadoc
- Outra documentação (e.g. Diagramas de classes)