Aula Prática 4

Resumo:

- Programação por Contrato.

Exercício 4.1

Pretende-se implementar um programa para o jogo "Advinha o número". Neste jogo, é secretamente escolhido um número aleatório num determinado intervalo ($[\min, \max]$), sendo o objectivo adivinhar esse número no menor número possível de tentativas. Por cada tentativa feita o jogo deve indicar se o número introduzido é o número certo (o que deverá fazer terminar o jogo), ou se é menor ou maior do que o que devia ser.

Para dar suporte à implementação deste jogo, implemente um módulo `GuessGame` com a seguinte interface pública:

- Construtor com dois argumentos (`min` e `max`) que inicializa o jogo com um número aleatório[1] no intervalo `[min,max]` (note que o intervalo não poderá ser vazio);
- Dois métodos inteiros – `min()` e `max()` – que indiquem os limites do intervalo definido para o objecto;
- Um método booleano – `validAttempt` – que indique se um número inteiro está dentro do intervalo definido;
- Um método booleano – `finished()` – que indique se o número foi já descoberto;
- Um método – `play` – que faça uma nova tentativa para adivinhar o número secreto. Este método só deverá ser aplicado a números que estejam dentro do intervalo definido, e enquanto o jogo não tiver terminado (ou seja, enquanto o número não tiver sido descoberto);
- Dois métodos booleanos – `attemptIsHigher()` e `attemptIsLower()` – que indiquem se a última jogada foi, respectivamente, acima ou abaixo do número secreto;
- Um método inteiro – `numAttempts()` – que indique o número de tentativas (jogadas) já realizadas.

[1] Pode utilizar o método `Math.random()` para gerar o número aleatório.

Note que, para a implementação do módulo, terá de escolher e definir o conjunto de atributos (que devem ser internos ao módulo) suficientes para a sua implementação (ou seja: que permitam saber os valores do intervalo, o número secreto, se o jogo terminou, se a última jogada é maior ou menor que o número secreto e o número de tentativas).

Para depuração do módulo, é fornecido um programa (`main`) embutido no próprio módulo, que utiliza a instrução `assert` para fazer alguns testes ao funcionamento do módulo[2].

Exercício 4.2

Utilizando o módulo do exercício anterior, crie um programa (`p42`) que implemente um jogo "Advinha o número", interagindo com o utilizador através de um `Scanner` sobre o teclado.

O programa pode receber dois argumentos indicando, respectivamente, os valores mínimo e máximo do intervalo, ou então não receber nenhum argumento assumindo o intervalo $[0; 20]$. Para a restante interface do programa, execute o ficheiro `jar` fornecido (`p42.jar`[3]).

Exercício 4.3

Modifique os módulos `Data` e `Nota` da aula anterior por forma a tentar assegurar a sua correcção[4]. Com esse objectivo, aplique contratos ao módulo `Data` por forma a garantir que os seus objectos correspondem sempre a datas válidas (para dar uma boa solução para este problema, considere a implementação dos métodos estáticos referidos no exercício 2.5). No caso do módulo `Nota`, aplique contratos de forma a garantir as seguintes propriedades em todos os objectos deste tipo:

- a data de fim da nota não pode ser anterior à data de inicio;
- todas as notas têm a si associadas um texto não vazio.

Exercício 4.4

No problema 3.5 da aula anterior a noção de "caixa" de dinheiro não tinha em consideração alguns aspectos muito importantes desta abstracção. Por uma lado, nada era dito sobre o valor possível das moedas. Por outro, sem uma aproximação apropriada visando correcção, nada impedia que a carteira tivesse, por exemplo, um valor negativo de dinheiro, ou que dela se retirasse mais dinheiro do que o lá existente. Corrija estas situações considerando que os valor correctos de moedas representam os pesos 1, 2 e 5 em todas as potências de 10 (1 cêntimo, 2 cêntimos, 5 cêntimos, 10 cêntimos, 20 cêntimos, 50 cêntimos, etc.).

[2]Note que o programa tem de ser executado com as asserções activadas.

[3]`java -ea -jar p42.jar`

[4]Para testar o módulo, copie o programa `p33.java` para `p43.java`.