Capítulo 07 - Definição de Classes

Exercício 06

Quais dos seguintes grupos de construtores são válidos?

```
public Cat(int age) { ... }
public Cat(double wgt) { ... }

public Dog(String name, double weight) { ... }
public Dog(String name, double height) { ... }

public Dog(String name, double weight) { ... }
public Dog(double weight, String name) { ... }

public Cat(String name) { ... }
public Cat(String name, double weight) { ... }
public Cat(double weight) { ... }
```

Exercício 12

Dada a classe abaixo

```
class Modifier {
    public static void change(int x, int y) {
        x = x - 10;
        y = y + 10;
    }
}
```

diga o que escreve o código seguinte

```
int x = 40;
int y = 20;
Modifier.change(x, y);
System.out.println("x = " + x);
System.out.println("y = " + y);
```

Exercício 13

Modifique a classe seguinte de forma a fazer parte de um *package* `myutil`.

```
class Person {
    private String name;
    public Person() {
        name = "Unknown";
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

Além de ajustar o código fonte, que passos são necessários de forma a que a classe fique acessível/utilizável por outras classes fora do *package* `myutil`?

Exercício 17

> Escreva uma aplicação para jogar o "Jogo de Fermi".

Gere aleatoriamente três dígitos secretos entre 0 e 9. Esses dígitos vão ocupar as posições 1, 2 e 3. O objetivo do jogo é **adivinhar no menor número de palpites os dígitos secretos e as respetivas posições**. Em cada tentativa o jogador indica um palpite para os dígitos que estão nas posições 1, 2 e 3. O programa responde com um código para cada dígito: `Fermi`, `Pico` ou `Nano`:

- Se o dígito está na posição correta, a resposta é `Fermi`.
- Se o dígito está noutra posição, a resposta é `Pico`.
- Se o dígito não está em nenhuma posição, a resposta é `Nano`.

Aqui estão alguns exemplos de respostas, supondo que os dígitos secretos são `6 5 8` nas posições `1 2 3` respetivamente:

Palpite	**Resposta**	Explicação
`1 2 5`	`Nano Nano Pico`	O valor 5 está na posição errada.
`8 5 3`	`Pico Fermi Nano`	O valor 5 está na posição correta. O valor 8 está na posição errada.
`5 8 6`	`Pico Pico Pico`	Todos os valores estão nas posições erradas.

Note que com as respostas dadas desta forma o jogador pode deduzir que dígitos não são "secretos". Por exemplo, na segunda resposta fica-se a saber que 3 não é secreto. **Para tornar o jogo mais interessante** ordene as

respostas ao acaso, ou alfabeticamente (por exemplo, a segunda resposta seria `Fermi Nano Pico` em vez de `Pico Fermi Nano`).

A aplicação joga repetidamente até que o jogador queira terminar. Depois de cada jogo mostra o número de tentativas.

Utilize comentários `Javadoc` para documentar as classes que desenhar para esta aplicação.

Exercício 18

> Escreva uma aplicação para (ajudar a) ensinar aritmética de frações a crianças.

Em cada **sessão** gere ao acaso dez **questões** que envolvam a soma, subtração, multiplicação e divisão de duas frações. No início de cada sessão o utilizador tem a opção de indicar o **tempo limite para responder às questões**. Se esse valor não for dado o valor por omissão é de 30 segundos. Depois de colocada a questão aguarde até que o utilizador responda. São **atribuídos pontos** de acordo com as regras seguintes:

Resposta	Tempo	Pontos
Correta	dentro do limite	10
Correta	além do limite	6
Errada	dentro do limite	3
Errada	além do limite	0

Quando terminar uma sessão use a consola para **escrever a classificação e o total de pontos** da seguinte forma:

```
                    Dentro do       Além do
                    Tempo Limite    Tempo Limite
Respostas Corretas  4               3
Respostas Erradas   2               1
PONTOS TOTAIS:      64 (40 + 18 + 6 + 0)
```

Depois de uma sessão terminar dê ao utilizador a opção de continuar para outra sessão.

Anexo: passos para a resolução do exercício 17

1. Identifique as **Tarefas** do Programa.
2. Escreva um **Documento de Desenho** com as **Descrições** das Classes.
3. Desenhe um **Diagrama do Programa**.

Além disso:

- Alinhe os **Passos de Desenvolvimento**.
- Proponha **Desenhos Alternativos** e justifique a sua escolha.
- **Teste** cada passo.
- Use o `Javadoc` para **Documentar** as classes.

Tarefas do Programa

Jogar o "Fermi" (mini MasterMind)

1. Três dígitos (0 … 9) aleatórios secretos.
2. Três posições a, b, c.
3. SEGREDO: atribuir os dígitos secretos às posições.
4. o JOGADOR tem de adivinhar o SEGREDO.
5. PALPITE: o JOGADOR indica três dígitos.
6. O PROGRAMA responde com um "CÓDIGO ^ 3".
7. CÓDIGO é `fermi` | `pico` | nano:
    - `fermi`: dígito OK + posição OK.
    - `pico`: dígito OK + posição ERR.
    - nano: dígito ERR.
8. A RESPOSTA baralha os CÓDIGOS.
9. Pede PALPITES até encontrar SEGREDO; Nessa altura mostra número de palpites.
10. Repete novo JOGO exceto se JOGADOR quer sair.

Fluxo Simples

1. Gerar Segredo.
2. Perguntar Palpite.
3. Avaliar Palpite.
4. Escrever Código.
5. Repetir 2. — 4. até que Palpite == Segredo.
6. Escrever Número de Palpites.
7. Perguntar Continuar.
8. Se Continuar, repetir desde 1.

Desenho

Classe	Propósito
`Secret`	Gera os dígitos do SEGREDO; Determina a RESPOSTA a um PALPITE.
`Guess`	Representa o PALPITE do Jogador.

Classe	Propósito
`Progress`	Mantém o registo do progresso no jogo: número de palpites; segredo atual; palpite atual;
`GameUI`	Interações com o jogador: pergunta o palpite; escreve a resposta; pergunta “continuar”

Diagrama do Programa

(exercício)