Programação Orientada por Objetos

2019/2020

Trabalho de Laboratório – Curso EI

1

Objetivos:
Composição de classes e coleções: revisões.

Programas:
Pretende-se desenvolver um programa que permita a gestão de uma frota de embarcações de pesca à rede.

Regras de implementação:
- Criar a aplicação utilizando o IDE BlueJ.
- Implementar o código necessário e testar no fim de cada nível.
- Use as convenções de codificação adotadas para a linguagem Java (ver **Notas**).

Implementação:

Nível 1:
- Implemente a classe `Tank`. Esta classe deverá ter como atributos:
  - `MAX_QUANTITY` – representa a quantidade máxima de peixe (kg) suportada no tanque, este valor deve ser constante e inicializado com 5000;
  - `id` – representa o identificador numérico do tanque, recebido como parâmetro;
  - `currentQuantity` – representa a quantidade atual de peixe (kg) existente no tanque, deve ser inicializada com o valor 0.
- Crie o construtor da classe `Tank` que recebe apenas o identificador numérico do tanque.
- Crie o método seletor da classe `Tank` para o atributo `currentQuantity`.
- Crie o método `isFull` que verifica se o tanque se encontra cheio e devolve um valor booleano.
- Crie o método `empty` que esvazia todo o peixe existente no depósito.
- Crie o método `insert` que recebe uma quantidade de peixe a adicionar no depósito. No caso de existir espaço disponível suficiente insere a quantidade recebida, caso contrário, insere apenas aquela que couber e devolve o valor da quantidade que sobrou.

Nível 2:
- Crie o enumerado `BoatStatus` que define os estados possíveis de uma embarcação: `FISHING` (a pescar), `MOORED` (atracado) e `SAILING` (a navegar).
- Implemente a classe `Boat`. Esta classe deverá ter como atributos:
  - `MAX_TANKS` – representa a quantidade máxima de tanques possíveis de instalar na embarcação, este valor deve ser constante e inicializado com o valor 6;
  - `numberOfFishermen` – representa o número de pescadores da embarcação;
  - `name` – representa o nome da embarcação;
  - `status` – representa o estado atual da embarcação.
  - `tanks` – `Arraylist` que guarda todos os tanques instalados na embarcação.
- Crie o construtor da classe `Boat`, que recebe como parâmetro os valores de todos os atributos exceto `tanks` e efetua as seguintes verificações:
  - `numberOfFishermen` – que deverá receber pelo menos 1 e não mais que 10. Caso contrário, atribuir o valor de 3 por *default*;
  - `description` – que não deverá conter uma String vazia. Caso contrário, atribuir o valor "desconhecido" por *default*;
  - `tankQuantity` – recebe a quantidade inicial de tanques, que não pode superar a quantidade máxima de tanques permitida. Caso contrário, atribuir a quantidade máxima. Deve em seguida criar e adicionar estes tanques á lista de tanques;
  - O estado inicial do barco é atracado (`MOORED`).
- Crie o método seletor da classe `Boat` para os atributos `numberOfFishermen` e `status`.

Nível 3:

- Para completar a classe **Boat**, comece por criar o método **installTank**, que caso exista espaço disponível, instala um novo tanque (**Tank**) na embarcação. Informa o utilizador se a operação foi concluída com sucesso ou não.
- Crie o método **startFishing**, que inicia o processo de pesca e altera o estado. Se a embarcação já se encontrar a pescar, informe o utilizador.
- Crie o método **returnToPort**, que retorna um barco ao porto e altera o estado. O barco só pode retornar se tiver as redes recolhidas. Informar o utilizador se não for possível.
- Crie o método **collect**, que termina a ação de pescar. Se a embarcação estiver em processo de pesca, gera um valor aleatório de peixe pescado (kg) compreendido entre [0, 20000]. Esta quantidade de peixe, será distribuída por todos os tanques com espaço disponível. Deve informar o utilizador a quantidade de peixe adicionada, se a capacidade máxima do barco for ultrapassada ou no caso de não ser possível recolher, pela embarcação não se encontrar em modo de pesca. No final, a embarcação passa ao estado de navegação.
- Crie o método **getTotalFish**, que retorna a quantidade total de peixe (kg) existente em todos os tanques da embarcação.

Nível 4:

- Implemente a classe **FishingFleet**. Esta classe deverá ter como atributos:
  - **COST_PER_BOAT** – representa o custo em euros de um barco ativo, este valor deve ser constante e inicializado com o valor 1200.0;
  - **COST_PER_WORKER** – representa o custo em euros de um trabalhador ativo, este valor deve ser constante e inicializado com o valor 60.5;
  - **GAIN_PER_KG** – representa o ganho em euros por quilo de peixe, este valor deve ser constante e inicializado com o valor 1.2;
  - **boats** – **HashMap** que armazena todos os barcos (**Boat**) da empresa, onde a chave é um código sequencial com o formato "B{1}".
  - **companyName** – representa o nome da empresa.
- Crie o construtor da classe **FishingFleet**, que recebe como parâmetro o atributo **companyName** que não deverá conter uma String vazia. Caso contrário, atribuir o valor "desconhecido" por *default*.
- Crie o método **registerBoat**, que recebe um barco e insere na coleção com uma chave sequencial.
- Crie o método **listBoats**, que lista todas as chaves dos barcos existentes na empresa.
- Crie o método **returnBoats**, que chama ao porto todos os barcos ativos (não atracados).
- Crie o método **returnBoat**, que recebe a identificação do barco e chama ao porto esse barco específico.
- Crie o método **sendBoat**, que recebe a identificação do barco e inicia a pesca desse barco específico.
- Crie o método **collectBoat**, que recebe a identificação do barco e recolhe as redes desse barco específico.
- Crie o método **getTotalWorkers**, que retorna o número total de trabalhadores da empresa.

Nível 5:

- Para completar a classe **FishingFleet**, de modo a ser possível obter informação relevante para a gestão, realize as alterações necessárias nos métodos **toString** das classes **FishingFleet**, **Boat** e **Tank** de forma a obter o seguinte output:

```
Empresa: Pescaria, Lda
Nº de trabalhadores: 9
Nº de embarcações: 2

Embarcações:
```

```
Barco: Caçador
        - 5 tripulantes
Tanques:
        - Tanque 1: 5000 kg
        - Tanque 2: 5000 kg
        - Tanque 3: 3063 kg
        - Tanque 4: 0 kg
        - Tanque 5: 0 kg
Capacidade: 13063/25000

Barco: Mar azul
        - 4 tripulantes
Tanques:
        - Tanque 1: 3870 kg
        - Tanque 2: 0 kg
        - Tanque 3: 0 kg
        - Tanque 4: 0 kg
Capacidade: 3870/20000

Contabilidade:
        - Despesa atual estimada: 2944.5€
        - Lucro atual estimado: 20319.6€
        - Balanço atual: 17375.1€
```

- Sugere-se a criação dos seguintes métodos auxiliares:
    - `getCurrentCost` – Calcula o custo atual de toda a frota ativa. Retorna o custo de todas as embarcações não atracadas e multiplica pelo seu custo, mais o valor do total de trabalhadores ativos multiplicado pelo seu custo.
    - `getCurrentGain` – Calcula o lucro atual de toda a frota ativa. Retorna o valor do total de peixe existente em todos os tanques multiplicado pelo valor do quilograma.

Notas:

Para os identificadores siga as convenções adotadas normalmente, em particular:

1) A notação **camelCase** para o nome das variáveis locais e identificadores de atributos e métodos.
2) A notação **PascalCase** para os nomes das classes.
3) Não utilize o símbolo '_', nem abreviaturas nos identificadores.