Instituto Superior de Engenharia de Coimbra

CTSP TPSI/DW

Programação Orientada a Objetos

Exercícios

# Ficha de exercícios 6

| | |
|---|---|
| Tópicos Abrangidos | • Herança e polimorfismo |

**Nota**: os exercícios desta disciplina e constantes nestas fichas destinam-se adquirir competência em programação orientada a objetos com os exercícios só funcionam se de facto os tentar fazer. Muito dificilmente conseguirá fazer a disciplina sem praticar. Não assuma que os docentes disponibilizarão soluções de forma sistemática. Deve tentar realizar os exercícios propostos e interagir com os docentes para esclarecer dúvidas e ouvir as soluções que forem propostas.

1. Imagine um conjunto de figuras geométricas, triângulos, retângulos, pentágonos, hexágonos, etc. Todas estas formas contém um diferente número de pontos com coordenadas x e y.
   Pense de que forma pode implementar classes que consigam descrever corretamente uma figura e seus diferentes tipos.
   Utilizando objetos, implemente a classe **Ponto**, a classe **Figura** e **três classes derivadas**.
   Discuta com o professor a melhor forma de resolver o problema, a estrutura das várias classes e os métodos a implementar.

2. Uma agência imobiliária tem, para vender, apartamentos e lojas comerciais. Os apartamentos são caracterizados pelo preço, área, andar e número de assoalhadas. As lojas são caracterizadas pelo preço e área, situando-se sempre no rés-do-chão. Qualquer destes bens imobiliários é caracterizado também por um código, gerado automaticamente, sendo formado pela palavra "apartamento" ou "loja", seguida por um inteiro que corresponde à ordem pela qual o objeto foi gerado (a sequência é a mesma para todos os bens). Os apartamentos têm um preço que é sempre 10 x área. As lojas têm um preço que é 15 x área. Tanto os apartamentos como as lojas devem permitir:
   - Obter o código
   - Obter o preço
   - Obter uma string com a descrição de todos os dados

   > O texto seguinte não faria parte do exercício se este aparecesse num exame. No contexto de exercício para treino, são acrescentadas algumas indicações que mais tarde é suposto nem ser preciso dizer.
   >
   > <u>Repare nos seguintes aspetos:</u>



CENTRO 2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Existem aqui dois conceitos muito relacionados entre si, mas ao mesmo tempo claramente distintos: apartamentos e lojas. Aquilo que partilham é relativamente óbvio. As diferenças que existem são tanto a nível de dados como e a nível de comportamento/restrições:

- Dados: um apartamento tem assoalhadas; uma loja não.
- Comportamento: uma loja está sempre no primeiro andar; um apartamento pode estar em qualquer andar.

A forma correta de modelizar esta situação em C# será a de fazer corresponder uma classe distinta a cada entidade Apartamento e Loja, e usar herança para fazer ambas as classes herdarem aquilo que partilham a partir de uma classe base comum: ambas as classes derivam de uma classe que encapsula os dados e comportamentos partilhados.

a) Defina as classes que representam estes bens dois bens imobiliários usando adequadamente os mecanismos de C# vocacionados para a reutilização de código e extensão de conceitos da forma conceito base – conceito especializado. Ajuda: vão ser necessárias três classes. Todas as classes garantir que os seus objetos só são construídos se lhe forem passados todos os dados relevantes.

b) Defina a classe Imobiliaria que representa (que armazena) todo o conjunto de bens imobiliários que a agência tem para vender neste edifício. Deve ser possível acrescentar bens imobiliários do edifício, fazer listagens de bens imobiliários por andar e pesquisa por código, obter os preços e descrição, e remover por código. Deve também ter em atenção que não deve utilizar uma estrutura de dados para cada tipo diferente de bem imobiliário (caso contrário o professor vai propor a existência de 50 tipos de imóveis) - podem mesmo existir muitos tipos diferentes – deve unificar o armazenamento de todos os bens imobiliários numa única estrutura de dados. A imobiliária deste exercício partilha algumas semelhanças com as imobiliárias da vida real:

- Não é a imobiliária que constrói os edifícios
- Se ocorrer um curto-circuito no sistema de informação da imobiliária, os edifícios continuam a existir.

c) c) Teste a funcionalidade da alínea anterior criando alguns imóveis, registando-os na imobiliária, e depois procurando-os/listando-os/etc.

Objetivos do exercício

- Treinar a identificação de situações em que é apropriado o uso de herança, a identificação e a definição das classes base e derivadas envolvidas.
- Treino das questões sintáticas envolvidas na herança e em particular, nos construtores. Acesso a dados privados da classe base a partir da classe derivada, e à funcionalidade em funções homónimas da classe derivada para a classe base (alínea a).
- Uso de polimorfismo (alínea b).

3. Considere o caso de Veículos. Pretende-se representar veículos através de classes em C#. No entanto, existem diversos tipos de veículos. Nomeadamente, existem automóveis, skates, bicicletas, aviões, navios, etc. Alguns têm

matricula, outros não. Alguns levam combustível, outros não. Alguns têm motor, outros não. Uns andam no ar, outros andam na água, outros na terra. No entanto, também é verdade que partilham comportamentos comuns. Por exemplo, deslocam-se, levam um certo número de passageiros, pertencem a alguém, etc.

a) Proponha uma hierarquia de classes para representar os veículos, definindo quais os dados e métodos que ficam em cada classe.

b) Implemente a classe base e pelo menos três classes derivadas. As duas classes devem ser razoavelmente completas, incluindo os construtores, que não deverão ser "os por omissão", métodos para obter dados, para modificar dados (os que fizer sentido que sejam modificados). Importante (próximos 6 pontos):

- Não é preciso guardar o "tipo" de veículo". Se isto lhe faz confusão, esclareça este ponto com o docente.
- Não deve usar dados protected. Este tipo de visibilidade não é tão mau como public, mas para lá caminha, e só em caso de justificada necessidade é que se deve usar (bem justificada, e este não é um desses casos).
- Não repita dados entre classe base e classe derivada.
- Não repita funções (funções a fazerem a mesma coisa) entre classe base e classe derivada. As classes derivadas devem usar as funções da classe base sempre que for necessário/útil.
- Inclua alguns mecanismos/funcionalidade que variam consoante o "tipo" de veículo, por exemplo: deslocar (a forma de deslocamento depende do "tipo" de veiculo), pagar imposto (alguns veículos são isentos, outros não, e a taxa varia de "tipo" para "tipo", obter a representação em cadeia de caracteres do veiculo ("tipos" diferentes têm dados diferentes), etc.
- Relativamente aos métodos (funções) nas classes, pondere cuidadosamente quais devem ser virtuais e nas classes derivadas, em quais é que usa new e em quais é que usa override. Verifique junto do docente a sua solução.

c) Na continuação da alínea anterior, repare que o veículo genérico (a classe base) terá mecanismos/funções em que não se sabe o que faz. Neste caso, a solução correcta é essa classe deixar os mecanismos em aberto, leia-se "por implementar". Verifique que a sua classe base está de acordo com isto. Verifique também que assim não poderá criar objectos dessa classe. Esclareça eventuais dúvidas quanto a este aspecto com o docente.

d) Implemente uma classe para alojar a função principal, nesta classe vai implementar de forma resumida um mecanismo para criar e adicionar veículos a uma colecção/matriz, consultar os dados (completos) do veículo (seja lá de que tipo for), e o também invocar a funcionalidade do veículo (por exemplo: deslocar, efectuar pagamento de imposto, etc.), em consonância com o que tiver previsto e implementado nessas classes em resultado das alíneas anteriores. Importante:

- O seu programa só pode ter uma colecção/matriz.

- O programa deve ser polimórfico: em nenhum momento o programa deve averiguar o "tipo" de veículo para saber o que deve fazer com ele (seja, que acção for: obter/imprimir dados, desencadear funcionalidade etc. Se contrariar este requisito, se esta fosse uma pergunta de exame, teria cotação O.
- A única excepção ao requisito anterior é o momento da criação do novo veículo. Nesse caso terá mesmo que perguntar ao utilizador o que deseja criar e obter os dados iniciais conforme o tipo de veículo e cria-lo. O mecanismo para acrescentar o novo veículo deve ser o mesmo para todos os tipos de veículo (ou seja, também é polimórfico).
- Se em algum momento tiver dúvidas quanto ao uso de polimorfismo (por exemplo, onde o deverei usar?), faça a si próprio a seguinte questão: "e nesta função, se houvesse 500 tipos diferentes de veículo? Fazia 500 versões diferentes desta função? Não pode ser…".

4. Pretende-se um conjunto de classes para lidar com os conceitos de Revistas, Quiosque e outros que surjam pelo meio.

a) Comece por definir a classe Revista, que tem por objetivo representar encapsular o conceito de revista.

- As revistas são constituídas por:
  - Um título;
  - Uma editora (nome)
  - A coleção a que pertence. Uma coleção é uma entidade com existência própria, definido pelo nome e pelo número da última revista que que saiu dessa coleção.
  - O número dentro da coleção a que pertence. Este número é atribuído pela coleção: sempre que asai" um novo número, a coleção faz avançar a numeração, gerando o número para a nova revista. A coleção sabe qual os títulos da coleção, não gerando novos números para títulos que já "saíram"
  - Um ISSN (um número do género ISBN que pode conter também letras).
- Pretende-se que os objetos da classe apenas possam ser inicializados com a informação relativa ao título, editora, a coleção e ISSN. A coleção já deverá existir previamente e fornecerá o número.
- Devem ser suportadas as seguintes operações Através de métodos):
  - Comparação de duas revistas:
    - Através de uma função para o efeito
    - Através do operador
    - Duas revistas são iguais se tiverem o mesmo título e se tiverem o mesmo número, independentemente de tudo o resto.
- Obtenção de uma cadeia de caracteres com toda a informação da revista.
- Saber se é apropriado para a idade x, sendo x fornecido e o resultado true ou false. A revista é apropriada para uma determinada idade conforme o tipo de conteúdo. A revista neste momento é genérica, não se sabendo o tipo de conteúdo e, portanto, este método, para já, não pode fazer nada.

b) Defina também as classes RevistaComics, BeiraDAgua, e RevistaParaColorir com as seguintes características:

- A classe RevistaComics representa uma revista como as descritas acima, mas com as seguintes diferenças/acrescentos:
- Nome do herói da história (cadeia de caracteres);
- Número de personagens que morrem na história;
- A revista é aconselhada apenas a partir da idade 12+número de personagens que morrem
- Obtenção dos dados como cadeia de caracteres: apresenta tudo o que é relativo à revista e ao que foi agora acrescentado. Se o número de personagens que morrem for maior que 10, em vez desta informação (o número) é obtido: "Só para mentes fortes".

c) A classe BeiraDAgua representa uma revista com generalidades acerca do tempo que vai fazer durante o ano, datas de cultivo, e meia dúzia de anedotas com mais de 500 anos cada uma. Tem as seguintes diferenças/acrescentos em relação a uma revista normal:

- Nunca pertence a nenhuma coleção. Esta informação é simplesmente ignorada/nulo. O número é sempre 1.
- O ano a que diz respeito.
- É apropriado apenas a pessoas com mais de 65 anos.

d) A revista para colorir é um conjunto de folhas com figuras a preto e branco para colorir. É apropriadíssima a crianças até aos 9 anos de idade e apenas a essas. Tem a informação acerca do nome da cada um dos desenhos e ainda permite acrescentar mais figuras (mesmo depois de impressas e à venda). Em tudo o resto é uma revista normal.

- Pretende-se também representar quiosques (os quiosques têm e vendem revistas).

e) Implemente a classe Quiosque sabendo que o quiosque:

- Tem revistas de toda a espécie e feitio.
- Permite acrescentar novas revistas, sabendo que o mecanismo que faz isto é apenas um, independentemente do número de tipos de revistas que existam.
- Permite apagar revistas dado o ISSN
- Permite obter a lista de revistas à venda. A lista é a descrição textual de todas as revistas existentes no quiosque.

5. Um ginásio decidiu informatizar toda a parte da gestão de clientes. Esta gestão envolve clientes, tarifários e outros aspetos. A descrição do problema é feita a partir dos aspetos mais pormenorizados para os mais gerais. É mesmo necessário ler o enunciado todo antes de começar a responder.

a) O montante a pagar por cada cliente do ginásio é determinado por um tarifário. Os tarifários são representados pela classe Tarifario que tem as seguintes características:

- Armazena um conjunto de inteiros que representam as durações dos treinos efetuados.
- Tem os métodos:
- acrescentaTreino. Recebe uma duração de treino feito e acrescenta à coleção.

- apagaTreinos. Apaga as durações de todos os treinos. Esta função não pode ser pública.
- calculaPagamento. Calcula o montante em dívida correspondente aos treinos armazenados, apaga os treinos e devolve o valor a pagar. O algoritmo que determina o valor a pagar não é conhecido nesta classe por isso este método não pode ser implementado.
- Implemente a classe Tarifário.

b) Existe um tarifário especial destinado a promover treinos curtos. O cálculo do custo neste tarifário é o seguinte:
    - cada treino até 10 minutos custa 10;
    - um treino entre 11 e 20 minutos custa 15;
    - um treino de 21 minutos ou mais custa 25.

Este tarifário é representado pela classe Apressado. Os objetos da classe Apressado são, naturalmente, também objetos de Tarifário. Existem outros tipos de tarifário, mas para este exercício apenas é necessário considerar este.

- Implemente a classe Apressado.

c) Implemente a classe Cliente que representa um cliente genérico do ginásio. Os clientes deste ginásio são de variados tipos, mas todos eles têm as seguintes características em comum:

- Têm nome e CC/BI.
- Têm um tarifário.
- Têm os seguintes métodos:
- iniciaTreino que recebe um inteiro que representa a hora em que começou um treino no ginásio. A hora é representada como o número de minutos a partir de um certo instante (mais pormenores na explicação acerca do ginásio). A hora é memorizada.
- terminaTreino que recebe o inteiro que representa a hora de saída do treino. A duração do treino é calculada e passada para um objeto Tarifario que o cliente tem.
- paga. Este método pergunta ao objeto Tarifario quanto é que é devido e esse valor é por sua vez devolvido por este método. A devolução do valor simboliza o pagamento pelos treinos efetuados desde o último pagamento.
- reageEntrada que reage à entrada de um cliente qualquer (outro cliente que não ele) no ginásio.
- Este método é genérico e não pode ser implementado já.
- reageSaida que reage à saída de um cliente qualquer (outro cliente que não ele) do ginásio. Este método é genérico e não pode ser implementado já.
- A criação de objetos deste tipo obriga à indicação (por parâmetro) do nome, BI e do objeto
- Tarifário associado ao cliente.
- Ainda acerca da classe Cliente há a dizer o seguinte:
- O cliente aqui descrito é um cliente genérico e estes comportamentos podem ser modificados em clientes mais específicos. A classe Cliente reflete este especto;
- Este cliente é tão genérico que tem métodos que nem se sabe por agora o que fazem (ex., reageEntrada, reageSaida). O melhor é que o código da classe seja feito de tal forma que impeça a criação de objetos.

d) Existem diversos tipos de cliente. Um deles é um tipo muito peculiar, com as seguintes características:

- Quando um (outro) cliente entra, não faz nada.
- Quando um (outro) cliente sai, verifica se está alguém (para além dele) no ginásio. Se não estiver, sai também, pois não gosta de estar sozinho.

Este cliente é representado pela classe Sociável. Implemente a classe Sociável tendo em atenção que se pretende que haja efetivamente objetos desta classe.

e) Por fim, pretende-se implementar o ginásio através da classe com o nome de Ginasio. As características do ginásio são as seguintes:

- Armazena um conjunto de clientes que podem ser de diversos tipos.
- Possui um relógio (para controlar os minutos a cobrar aos clientes). O relógio segue uma novíssima norma internacional simplificada: começa em zero, tem incrementos de I e não tem segundos nem horas. É apenas um inteiro que começa em zero e aumenta de I em I (se achar que este modelo de relógio não é realista, pode implementar um relógio mais complexo, mas o tempo dado para fazer o exercício é o mesmo).

O ginásio tem mecanismos (métodos) para:

- Fazer passar um certo número de minutos.
- Acrescentar um cliente ao ginásio. O cliente é previamente construído e passado ao método. Este método é compatível com qualquer tipo de cliente.
- Remover um cliente, dado o CC/BI.
- Entrada de um cliente no ginásio — o cliente indicado pelo BI passa a treinar (entra na sala de treino). Todos os clientes a treinar no ginásio (subconjunto dos que existem ao todo no ginásio) são notificados que este cliente entrou. O cliente é informado da hora a que entrou. O cliente é especificado por BI e terá que existir previamente no ginásio.
- Saída de um cliente —o cliente indicado pelo BI termina o treino e sai do ginásio (da sala de treino). Todos os clientes restantes a treinar no ginásio são notificados que este cliente saiu. O cliente é informado da hora a que saiu. O cliente é indicado por BI e terá que existir previamente no ginásio.
- Pagamento da conta de um cliente, dado o CC/BI.

Eventualmente, na resolução das alíneas deste exercício, pode ser necessário acrescentar mais dados ou métodos às classes. Se achar que isso acontece, acrescente e assinale a razão.

Nota: Este exercício é um bom exemplo do que pode sair num teste nesta disciplina. A única diferença seria que, no teste, não seriam dadas tantas ajudas (exemplo de ajuda: "vai haver esta que função que faz assim", etc.) Veja se realmente resolve o exercício na aula para praticas.