ESQUEMA AULA PRÁTICA 11

❑ Herança e Classes Abstractas

Uma classe é ***abstracta*** se pelo menos um dos seus métodos de instância não se encontra implementado (só declarado sintacticamente). A primeira implicação é que uma tal classe não poderá ser instanciada. Porém o mecanismo de herança ainda vigora. Sendo assim qual a utilidade?

- Normalização e consequente redução de vocabulário: todas as subclasses concretas de uma classe abstracta terão que implementar os métodos que foram declarados como abstractos na classe abstracta.
- Todas as futuras implementações de subclasses de uma classe abstracta utilizarão uma linguagem comum. O polimorfismo garantirá que a resposta dada por uma qualquer instância de uma subclasse à mensagem recebida é a apropriada.

1. Defina uma classe abstracta, `Count`, para representar contadores, com operações para incrementar de uma unidade, decrementar de uma unidade, inspeccionar o valor corrente e reinicializar a zero.

   - Defina classes derivadas `CountPositive` em que a operação para decrementar não tem efeito se o contador estiver a zero e `CountModulus` que volta a zero quando atinge um valor máximo predeterminado, e volta a esse valor máximo menos um ao decrementar zero.

2. Estude as API da classe abstracta `Number` e das suas subclasses `Double`, `Float`, `Integer` e `Long`. Faça um programa para testar os métodos disponibilizados.

3. Estude a API da classe abstracta `AbstractCollection` presente no *package* `java.util`. Apresente em esquema a hierarquia de classes que dela resulta. Faça um programa para testar os métodos disponibilizados em uma das concretizações da referida classe.

4. Pense no conjunto de mensagens a que uma qualquer figura bidimensional (e.g. ponto, linha, círculo, elipse, triângulo, uma qualquer área poligonal fechada, etc.) deverá ser capaz de responder. Defina a classe (abstracta?) `Figura2D` e programe as subclasses sugeridas e outras que considere pertinentes.

Exercícios extra aulas:

5. Simplificando, podemos afirmar que, uma pessoa é alguém de quem sabemos o nome, o sexo e a nacionalidade. Programe a classe `Pessoa` com os construtores (entre eles implemente o construtor não parametrizado – qual a utilidade? – e o *construtor por cópia*), selectores e modificadores que considerar necessários bem como os métodos públicos `toString`, `clone` e `equals` (ainda que os considere pouco naturais para uma pessoa terão interesse do ponto de vista da programação!).

6. Programe a classe `Autor` de obra literária. Um autor é uma pessoa sobre a qual será interessante saber o ano de nascimento e falecimento (caso já tenha falecido), a sua nacionalidade bem como o prémio literário mais importante que conquistou, se detivermos esta informação.

7. Programe a classe `Musico`. Um músico pode ser um indivíduo ou uma banda de que interessa manter uma descrição e o seu estilo musical dominante.

8. Programe a classe `Amigo`. Um amigo é uma pessoa de quem sabemos a data de nascimento, o ano em que o conhecemos, um contacto, o nível de amizade que por ela nutrimos e ainda o "parceiro" com quem normalmente "anda".

9. Pretende-se organizar a colecção de livros e CDs áudio das nossas biblioteca e discoteca particulares[1].
Um livro é caracterizado pelo título, pelo autor, pela língua em que está escrito, pelo seu valor afectivo e se já o lemos pela indicação da data em que concluímos a sua leitura. Por vezes emprestamos livros a amigos e é necessário saber se o livro está emprestado e a quem.
A nossa colecção de CDs de música tem um tratamento semelhante à dos livros: podemos emprestá-los, têm determinado valor afectivo e poderemos querer saber quando os ouvimos pela última vez.
Existem pois analogias entre as duas classes e também diferenças (por exemplo, o CD de música não tem autor mas sim músico/banda e não será necessário indicar a língua mas sim o estilo de música).
Agrupe as semelhanças de comportamento e estrutura numa classe chamada `Biblionimo`[2]. O objectivo desta classe é simplificar a subsequente programação das classes `Livro` e `CDAudio`.

10. Programe uma classe (`FilmeDVD`) para manter e manipular informação sobre os filmes de uma filmoteca em suporte DVD.

[1] Exercício adaptado de "Programação com classes em C++", *P. Guerreiro*, FCA, 2000.

[2] Bibliónimo significa nome de qualquer livro de reputação universal (do grego *bíblion* (livro) + *ónyma* (nome)). Parece ser por isso um nome apropriado (até que sugiram um melhor) para uma classe que procura disponibilizar o comportamento comum dos nossos objectos culturais de consumo.