Programação Orientada a Objetos

Prof. Paulo Henrique Pisani

Prof. Saul de Castro Leite

http://professor.ufabc.edu.br/~paulo.pisani/

UFABC

junho/2018

Exercício 1 – Figuras Planas

Ricardo Drudi (PRAE)

Você ficou encarregado de criar um sistema para calcular a área e o perímetro de **figuras planas convexas**. Seu trabalho é:

- Criar uma hierarquia de classes que inclua **círculo**, **triângulo**, **quadrado** e **retângulo**.
- As figuras triângulo, quadrado e retângulo devem ser representadas pelo comprimento de seus lados. O círculo será representado pelo seu raio.
- Crie métodos para retornar as medidas de área e perímetro (note que para o triângulo, é possível calcular a área conhecendo apenas seus lados com a *fórmula de Heron*).
- Sobrecarregue o método toString() para mostrar informação detalhada sobre cada figura (nome, área e perímetro).

UFABC

Atenção: todas as figuras devem estar no **pacote geometria**.

Exercício 1 – Figuras Planas

Crie um programa que tenha um vetor de figuras planas. Nesse vetor, instancie uma figura de cada tipo e imprima na tela as informações de cada figura.

O programa ficará no pacote “criafiguras”.

Exercício extra: inclua uma validação das entradas das figuras. Se o usuário passar argumentos inválidos, deve ser lançada uma exceção.

UFABC

Exercício 2 – Prova errada

- O professor ABC resolveu escrever um programa para usar na prova (link "**CodigoProvaErrada**" no site). Entretanto, o código está com vários erros... Você poderia ajudar o professor a corrigir seu programa?
- A saída do programa deveria ser essa:

```
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: Quanto eh 1 + 1?
- 80
- 2
- 700
Enunciado: int[] v = new int[3] cria um vetor com 3 nulls
[ ] V ou F
```

UFABC

Exercício 2 – Prova errada

- O professor ABC resolveu escrever um programa para usar na prova (link "**CodigoProvaErrada**" no site). Entretanto, o código está com vários erros... Você poderia ajudar o professor a corrigir seu programa?
- A saída do programa deveria ser essa:

```
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: Quanto eh 1 + 1?
- 80
- 2
- 700
Enunciado: int[] v = new int[3] cria um vetor com 3 nulls
[ ] V ou F
```

Importante: não altere os modificadores de acesso! E não adicione atributos!

UFABC

Faça apenas mudanças pontuais, não é preciso reestruturar o programa!

Exercício 3 – Time de futebol

- Crie uma hierarquia de classes para o seguinte problema (no pacote copadomundo):
  - Um time de futebol possui vários jogadores, que podem ser: atacante, lateral ou goleiro.
  - Todo jogador possui um número na camisa (definido pelo construtor) e um método para imprimir sua posição (e.g. atacante, goleiro, etc). Um lateral pode ficar na direita ou esquerda do campo (isso é definido por parâmetro no construtor).
- No time de futebol, haverá um vetor de jogadores. Esse vetor deve ser armazenado em ordem crescente de número da camisa;

UFABC

Exercício 3 – Time de futebol

- Um time de futebol possui um método para **substituir** jogador:
  - Este método deve fazer a substituição e manter o vetor de jogadores ordenado por número da camisa.
- Para este exercício, vamos considerar que um time pode ter entre 1 e 11 jogadores.
- Adicione um método para imprimir a escalação atual também.

UFABC

Exercício 3 – Time de futebol

- No programa principal (que ficará no pacote default):
  - Leia 3 jogadores e insira no Time ABC
  - Leia outros 3 jogadores e insira no Time DEF
  - Imprima as duas escalações
  - Leia o número de um jogador (que existe em ABC) e outro que entrará (como substituição) no time
  - Faça a substituição e imprima as duas escalações

- **Exercício extra**: crie um método verificaTime(), que retorna true se o time é válido (ou seja, 11 jogadores, sendo apenas 1 goleiro).

UFABC

Exercício 4 (extra) – Figuras ligadas

Usando as classes do exercício anterior, o objetivo deste exercício é criar uma **lista ligada** de figuras planas. Esta lista ligada deve conter um método que permite a **inserção ordenada crescente** das figuras geométricas, respeitando a seguinte ordem:

- os objetos são ordenados de acordo com o número de vértices que os compõe. Define-se o círculo como a maior figura por este critério. Quando os objetos empatam em número de vértices, usa-se como critério de desempate, o tamanho da área e em seguida o tamanho do perímetro. Isto é, se duas figuras são retângulos, então definimos a ordem por suas áreas e em caso de empate, pelo perímetro.
- Para que a lista ligada saiba comparar as figuras, implemente um método (na classe FiguraPlana) que recebe como argumento um objeto FiguraPlana e retorna -1 caso a figura receptora da chamada seja menor que a figura recebida por argumento, 0 caso sejam iguais e 1 se a receptora for maior.
- A lista ligada deve ter um método que imprima na tela as figuras na ordem que aparecem na lista.

Implemente um programa principal para testar esta ListaLigada.

UFABC