Universidade Federal de Mato Grosso

Bacharelado em Ciência da Computação

Semestre: 2024/2

Programação 3

Lista 3

Exercício 1: Formas Geométricas

Desenvolva um sistema para modelar formas geométricas. Siga as especificações abaixo:

1. Crie uma classe abstrata `FormaGeometrica` com os seguintes elementos:
   - Um método abstrato `double calcularArea()` que retorna a área da forma.
   - Um método concreto `double calcularPerimetro()` que retorna o perímetro, inicializado como `0.0`.
   - Sobrescreva o método `toString()` para exibir informações sobre a forma.
2. Crie duas classes derivadas:
   - `Circulo`: Implementa `calcularArea()` usando πr^2 e sobrescreve `calcularPerimetro()` como $2\pi r$.
   - `Retangulo`: Implementa `calcularArea()` como `base`$\times$`altura` e sobrescreve `calcularPerimetro()` como $2 \times$ (`base` + `altura`).
3. Para ambas as classes derivadas, sobrescreva o método `equals` para retornar verdadeiro quando os atributos foram iguais, mas garantindo que antes desta comparação sejam feitas as verificações de igualdade de objeto e de classe demonstradas na sobrecarga do `equals` no último slide da aula.
4. Sobrescreva o método `toString()` em ambas as classes para exibir as dimensões, área e perímetro.
5. No método principal:
   a) Crie um array de objetos `FormaGeometrica` contendo instâncias de `Circulo` e `Retangulo`.
   b) Percorra o array e imprima as informações de cada objeto usando `toString()`.
   c) Calcule e exiba as áreas e perímetros.
   d) Crie dois objetos com as mesmas dimensões e compare-os usando `equals()`.

Exercício 2: Personagens de RPG

Crie um sistema para modelar personagens de um RPG. Siga as especificações abaixo:

1. Crie uma classe base `Personagem` com os seguintes elementos:
   - Atributos:
     - `String nome` (privado, acessível por métodos `get` e `set`).
     - `int nivel` (protegido).
     - `int pontosDeVida` (protegido, inicia com 100).
   - Métodos:

  - `void exibirDetalhes()` (imprime o nome, nível e pontos de vida do personagem).
  - `void receberDano(int dano)` (reduz os pontos de vida em `dano`, mas não deixa o valor ficar abaixo de 0).
- Construtor para inicializar os atributos `nome` e `nivel`, com `pontosDeVida` fixado em 100.

2. Crie as seguintes classes derivadas:

- `Guerreiro`:
  - Atributos:
    - `int forca` (protegido, representa o dano causado por ataques).
  - Métodos:
    - Sobrescreva `exibirDetalhes` para incluir a força.
    - `void atacar(Personagem inimigo)` (causa dano equivalente à força ao inimigo chamando o método `receberDano()`).
  - Construtor para inicializar `nome`, `nivel` e `forca`.
- `Mago`:
  - Atributos:
    - `int mana` (protegido, inicia com 50, representa energia para lançar magias).
    - `int poderMagico` (protegido, representa o dano causado por magias).
  - Métodos:
    - Sobrescreva `exibirDetalhes` para incluir mana e poder mágico.
    - `void lançarMagia(Personagem inimigo)`:
      - Causa dano equivalente a `poderMagico`.
      - Reduz a mana em 10 por magia lançada.
      - Não permite lançar magia se a mana for menor que 10.
  - Construtor para inicializar `nome`, `nivel`, `mana` e `poderMagico`.
- `Arqueiro`:
  - Atributos:
    - `int precisao` (protegido, representa a chance de acerto de 0 a 100).
  - Métodos:
    - Sobrescreva `exibirDetalhes` para incluir a precisão.
    - `void dispararFlecha(Personagem inimigo)`:
      - Utiliza um número aleatório (de 1 a 100) para determinar se o ataque foi bem-sucedido (acima de 50, por exemplo).
      - Causa dano fixo de 20 pontos se o disparo for bem-sucedido.
  - Construtor para inicializar `nome`, `nivel` e `precisao`.

3. No método principal:

a) Crie pelo menos 3 personagens, um de cada classe derivada (`Guerreiro`, `Mago` e `Arqueiro`).

b) Inicialize os atributos de cada personagem no momento da criação.

c) Imprima os detalhes de todos os personagens.

d) Simule uma batalha onde:

- O `Guerreiro` ataca o `Mago`.
- O `Mago` lança uma magia no `Arqueiro`.
- O `Arqueiro` tenta disparar uma flecha contra o `Guerreiro`.

e) Após cada ação, exiba os detalhes atualizados dos personagens.