UPskill – Java+.NET

Programação Orientada a Objetos – Herança e Polimorfismo

UPskill
Digital Skills & Jobs

P.PORTO

EXERCÍCIO JOGO PEDRA-PAPEL-TESOURA

Neste exercício, o objetivo é criar um jogo simples de pedra-papel-tesoura, no qual os elementos do jogo serão modelados através da implementação de uma hierarquia de classes utilizando classes abstratas, herança e polimorfismo.

1. Defina uma classe abstrata denominada GameElement com os seguintes membros:
    a. Crie a propriedade ElementName, do tipo string, representando o nome do elemento do jogo (por exemplo, Pedra, Papel, Tesoura).
    b. Crie a propriedade Player, do tipo inteiro, representando o jogador que possui este elemento de jogo.
    c. Implemente um construtor com modificador de visibilidade protected. Este construtor deve possuir dois parâmetros, elementName e player, atribuindo esses valores às propriedades ElementName e Player, respectivamente.
    d. Crie o método Draw na classe abstrata GameElement. Este método deve receber um argumento do tipo GameElement e retornar um valor booleano indicando se os dois elementos são do mesmo tipo.
    e. Sobrescreva o método ToString. Este método deve retornar uma string identificando o elemento e o jogador ao qual pertence.
    f. Implemente um método chamado DisplayDescription, permitindo a apresentação da descrição do elemento na consola.
    g. Inclua um método abstrato chamado Defeat. Este método deve receber um argumento do tipo GameElement e cada classe derivada de GameElement deve implementar esse método, retornando um valor booleano que indica se o elemento atual vence sobre o elemento passado como parâmetro.
2. Criar os elementos de jogo.
    a. Criar as classes concretas que herdam de GameElement para cada elemento:
        i. Rock (Pedra): Um elemento de jogo de pedra que derrota a tesoura.
        ii. Paper (Papel): Um elemento de jogo de papel que vence a pedra.
        iii. Scissors (Tesouras): Um elemento de jogo de tesoura que derrota o papel.
    b. Cada uma destas classes deverá ter um construtor que recebe apenas um único parâmetro para o identificador do jogador. Este construtor deverá utilizar o construtor da

classe abstrata GameElement, utilizando um nome por defeito que identifique o tipo específico do elemento.

c. Implemente o método Defeat segundo as regras acima descritas.

d. Sobrescreva o método DisplayDescription em cada uma das classes concretas (Rock, Paper e Scissors) para proporcionar descrições mais apropriadas para cada elemento, por exemplo, "Cutting scissors with two blades" para a classe Scissors

e. Sobrescreva o método ToString para que, antes da representação textual obtida da classe GameElement, seja exibida uma representação gráfica do elemento. Exemplo para a classe Scissors:

3. Desenvolva um programa que permita a dois jogadores enfrentarem-se neste jogo. Primeiro, será necessário especificar o número de jogadas a serem efetuadas. Em seguida, para cada jogada, os jogadores devem selecionar o elemento do jogo desejado, com validação adequada dos inputs do utilizador. Após a escolha dos elementos por ambos os jogadores, as jogadas são realizadas, e os resultados de cada rodada devem ser apresentados na consola
4. No final, apresente um resumo das jogadas, indicando se houve empate, vitória ou derrota para cada jogador/jogada. O jogador com mais vitórias deve ser destacado no final deste resumo.
5. Crie uma classe para um papel especial (SpecialPaper). Este elemento de jogo apenas difere do Paper pela sua habilidade de vencer todos os outros elementos (exceto o Paper), sendo que o Paper ainda consegue vencer todas as classes derivadas desta.

   Dica: o1.GetType().IsSubclassOf(o2.GetType()) retorna um valor booleano a indicar se o1 é de uma subclasse de o2.
6. A opção de escolher esta nova classe como elemento de jogo deve ser aleatoriamente (com uma probabilidade de 30%) atribuída aos jogadores no momento da seleção dos elementos.