Computação I

Lista de exercícios 11 – Introdução à Orientação a Objetos
Atenção! Leia as instruções antes de fazer a lista! - Data de entrega: 14/12/2022
Não é necessário testar se os dados passados por argumento são válidos a menos que pedido na questão. Como de costume, os métodos devem ter os nomes das questões (questao2, questao3, questao4...), exceto pelo método construtor (que tem um nome especial __init__), pela questão 6, que deve criar uma classe com o nome personagem2, e pelo método da questão 8, pois nessa questão é necessário sobrescrever um método herdado. Para que isso funcione o método da classe herdeira deve ter mesmo nome do método da classe herdada (no caso, questao4). Ao definir os métodos preste atenção para o fato de que, além dos argumentos pedidos, um método também recebe um argumento que é uma referência ao objeto (esse argumento é, necessariamente o primeiro argumento do método, ele é um argumento implícito ao fazer a chamada do método e, em geral, ele é chamado de self). Não crie atributos além daqueles pedidos na questão 1: é permitido usar variáveis de auxílio dentro dos métodos, mas essas variáveis não devem ser atributos (o atributo está associado ao objeto e necessita do self).

Crie uma classe para representar um personagem de um jogo para capturar pokemons. O nome da classe deve ser personagem. Essa classe utilizará a função randint da biblioteca random, faça o import antes de definir a classe. Considerando essa classe, defina os seguintes métodos:

1. Método construtor, que define os atributos: nome; x e y, que representam a posição do jogador no cenário; experiência; e pokemons (os nomes dos atributos devem ser: nome, x, y, experiencia e pokemons – atenção a esses nomes para auxiliar a correção! Todos as letras devem estar em minúscula!). O atributo que guarda o nome é uma string e essa string deve ser passada por argumento. Os atributos x e y devem valer 0, experiência deve valer 2, e pokemons deve ser uma lista vazia (esses quatro valores são arbitrários e não devem ser recebidos por argumento).

2. Um método de andar, que recebe por argumento uma string com o sentido (“cima”, “baixo”, “direita” ou “esquerda”) que o personagem deve ser movimentar. Se a string passada por argumento for “cima”, o atributo y deve ser somado em 1 (isto é, o novo valor desse atributo será igual ao que havia antes mais 1); se a string for “baixo”, o atributo y deve ser subtraído em 1 (isto é, o novo valor desse atributo será igual ao que havia antes menos 1); se a string passada por argumento for “direita”, o atributo x deve ser somado em 1 (isto é, o novo valor desse atributo será igual ao que havia antes mais 1); se a string for “esquerda”, o atributo x deve ser subtraído em 1 (isto é, o novo valor desse atributo será igual ao que havia antes menos 1). Se a string não for nenhuma dessas opções, simplesmente não modifique os atributos. Em qualquer uma dessas situações, o método deve retornar o valor do atributo x e o valor de atributo y.

3. Um método de treino, para modificar a quantidade de experiência de acordo com um valor sorteado (esse método não recebe nenhum argumento além do self, necessário para que o Python tenha uma referência para o objeto). O método deve utilizar a função randint do módulo random para sortear um número entre 0 (inclusive) e 5 (inclusive). Se o valor sorteado somado ao valor que já havia no atributo de experiência for maior que 100, então o atributo deve ser mantido em 100. Caso contrário, o valor do atributo de experiência deve ser igual ao que já havia no atributo somado ao valor sorteado. O método deve retornar o novo valor do atributo de experiência.

4. Um método para tentar capturar um pokemon, que recebe por argumento uma string com o nome do pokemon e um inteiro com a quantidade de *defesa* desse pokemon. O método deve usar a função randint do módulo random para sortear um número entre -1 (inclusive) e 3 (inclusive) que indica a *força* do personagem ao tentar capturar o pokemon. Se a experiência do personagem (guardada no atributo) somada à *força* sorteada for maior que a *defesa* do pokemon (recebida por argumento), então o

pokemon foi capturado. Nesse caso, o nome do pokemon (string recebida por argumento) deve ser incluído ao atributo que contém a lista de pokemons e o método deve retornar 1. Caso contrário, nenhum atributo deve ser modificado e o método deve retornar 0.

5. Um método que verifica se você está próximo do pokemon antes de tentar captura-lo. O método deve receber por argumento uma string com o nome do pokemon, um inteiro com a quantidade de defesa desse pokemon, um inteiro com a posição x do pokemon, e um inteiro com a posição y do pokemon. Se a posição x do personagem menos a posição x do pokemon for maior ou igual a -1 e menor ou igual a 1 e se a posição y do personagem menos a posição y do pokemon for maior ou igual a -1 e menor ou igual a 1, então o método da questão 4 deve ser chamado. O valor de retorno desse método deve ser a posição do personagem menos a posição x do pokemon e a posição y do personagem menos a posição y do pokemon.

6. Crie a classe personagem2, herdeira de personagem. Considerando essa nova classe, defina os métodos das questões 7 e 8.

7. Um método que recebe uma string com o nome de um pokemon por argumento e retorna a quantidade de vezes que esse pokemon aparece no atributo do tipo lista que contém todos os pokemons do personagem.

8. Sobrescreva o método de tentar capturar um pokemon (questão 4, atenção ao nome desse método) de forma que o método de tentar capturar um pokemon nova classe (personagem2) funcione da seguinte forma: recebe por argumento uma string com o nome do pokemon e um inteiro com a quantidade de defesa desse pokemon; sorteia um valor inteiro maior ou igual a 1 e menor ou igual à experiência do personagem (atributo de experiencia), onde o valor sorteado é a *força*; sorteia um valor inteiro maior ou igual a 1 e menor ou igual à quantidade de defesa do pokemon (recebida por argumento), onde o valor sorteado é o *poder de defesa*. Se a *força* for maior que o *poder de defesa*, então o pokemon foi capturado, o nome do pokemon (string recebida por argumento) deve ser incluído ao atributo que contém a lista de pokemons e o método deve retornar 1. Caso contrário, nenhum atributo deve ser modificado e o método deve retornar 0.