MAB225 – Computação II – Aula Prática 2

Objetivo: praticar os conceitos iniciais de programação orientada a objeto.

Para todos os exercícios abaixo: teste as classes: crie instâncias, modifique os atributos e utilize os métodos.

1. Crie uma classe para representar uma agenda de compromissos. A classe deve possuir os seguintes atributos e métodos:

    a. Atributo de lista de compromissos – lista de tuplas onde cada tupla possui um inteiro que representa o ano, um inteiro para o mês, um inteiro para o dia e uma string com a descrição do compromisso.
    b. Método para adicionar um compromisso – recebe como argumentos o ano, mês, dia e descrição, cria a tupla e a adiciona ao atributo de lista de compromissos. Verifique se os valores passados por argumento são dos tipos corretos.
    c. Método para buscar compromissos – recebe ano e mês ou ano, mês e dia, e retorna uma lista de tuplas com todos os compromissos daquele mês ou com todos os compromissos daquele dia, dependendo da entrada do usuário. Defina o dia com argumento padrão igual a zero. Verifique no código se o dia é igual a zero. Caso afirmativo, o usuário não passou o argumento de dia e deseja buscar todos os compromissos de um mês. Caso contrário, ele deseja buscar os compromissos de um dia específico.
    d. Método para cancelar um compromisso – recebe como argumentos ano, mês e dia. Utiliza o método da letra (c) para buscar todos os compromissos daquele dia e lista os mesmos de forma sequencial e numerada. Em seguida, pergunta ao usuário (input) qual o número do compromisso que deseja cancelar. Se o usuário digitar um número que não está na lista, imprima uma mensagem dizendo que nenhum compromisso foi cancelado. Caso contrário, remova o compromisso escolhido da lista de compromissos. (Dica: utilize o método remove do tipo lista. Como argumento do remove, utilize a tupla com ano, mês, dia e descrição do compromisso.)

2. Crie uma classe para representar um personagem de um jogo. Essa classe utilizará a função randint da biblioteca random, faça o import antes de definir a classe. O personagem deve possuir os seguintes atributos e métodos:
    a. Dois atributos do tipo string com o nome e o país de nascimento do personagem. Ambas strings devem ser definidas inicialmente como strings vazias.
    b. Três atributos do tipo inteiro com ponto de vida, força de ataque e força de defesa. Ponto de vida deve começar valendo 100, ataque valendo 5 e defesa valendo 7.
    c. Um método para modificar os atributos de nome e país. O método não recebe essas informações como argumentos, ele deve utilizar a função input para perguntar ao usuário o nome e o país (separadamente) e depois adicionar aos atributos.

d. Um método de apresentação, que imprime o nome e nacionalidade do personagem em uma frase: “Olá, meu nome é *fulano* e eu sou do *país*.”, onde fulano e país devem ser substituídos pelos valores do atributo.
e. Um método de treino, que recebe uma string com a habilidade (“ataque” ou “defesa”) que deseja treinar. O método deve sortear um número entre 0 e 3 (utilize a função randint importada) a acrescentar o valor sorteado ao atributo de ataque ou defesa dependendo da string passada por argumento. Se a string passada não for nenhuma das duas válidas, imprima uma mensagem de erro.
f. Um método para modificar a quantidade de pontos de vida que recebe um inteiro positivo ou negativo como argumento. O valor passado por argumento deve ser acrescentado ao valor que já existia no atributo de pontos de vida. Respeitando a condição que o atributo pontos de vida deve estar entre 0 e 100.
g. Um método para recuperar pontos de vida. Esse método utiliza randint para sortear um número entre 1 e 5 e utiliza o método da questão (f) para acrescentar o valor sorteado ao argumento com os pontos de vida.
h. Um método para atacar que recebe como argumento um outro objeto do tipo personagem. Verifique se o argumento passado é do tipo desejado (dica: utilize a função type e compare o tipo do próprio objeto – self – com o tipo do argumento; os tipos devem ser iguais). Sorteie um valor entre 1 e 5 para o dano causado ao atacar. Se o dano somado ao atributo de ataque for maior que a quantidade de defesa do objeto passado por argumento, então o ataque foi bem-sucedido e o método da questão (f) do objeto passado por argumento deve ser chamado considerando que o dano deve ser subtraído da quantidade de pontos de vida do objeto passado por argumento. Imprima uma mensagem indicando o sucesso do ataque. Caso contrário (o dano somado ao atributo de ataque é menor que a quantidade de defesa do objeto passado por argumento), imprima uma mensagem informando que o ataque falhou.