FIAP

FIAP

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DISRUPTIVE ARCHITECTURES: IOT, IOB & IA

01 – Orientação a Objetos com Python

Prof. Airton Y. C. Toyofuku

profairton.toyofuku@fiap.com.br

Exercícios

1. Crie uma classe "Pokemon" que possua atributos como "nome", "tipo", "ataque" e "defesa". Adicione métodos para retornar esses atributos e sobrecarregue o operador de comparação para permitir a comparação de dois pokemons com base em seu ataque.
2. Crie uma classe "Treinador" que herde da classe "Pessoa" e adicione um atributo "time" que será uma lista de pokemons. Adicione métodos para adicionar e remover pokemons do time.
3. Crie uma classe "Batalha" que possua dois treinadores e determine o vencedor da batalha com base nos pokemons de seus times.
4. Crie uma classe "Pokebola" que possua um pokemon capturado. Adicione métodos para liberar o pokemon e trocar o pokemon capturado.
5. Crie uma classe "Loja Pokemon" que venda itens para treinadores, como poções e pokebolas. Adicione métodos para adicionar e remover itens da loja e para realizar compras.
6. Crie uma classe "Pokedex" que possua informações sobre os pokemons, como ataques, defesas e tipos. Adicione métodos para adicionar pokemons na pokedex e pesquisar informações sobre um pokemon específico.

Resolução Exercício 1

FIAP

```python
class Pokemon:
    def __init__(self, nome, tipo, ataque, defesa):
        self.nome = nome
        self.tipo = tipo
        self.ataque = ataque
        self.defesa = defesa

    def get_nome(self):
        return self.nome

    def get_tipo(self):
        return self.tipo

    def get_ataque(self):
        return self.ataque

    def get_defesa(self):
        return self.defesa

    def __lt__(self, other):
        return self.ataque < other.ataque
```

Resolução Exercício 2

```python
class Pessoa:
    def __init__(self, nome):
        self.nome = nome

class Treinador(Pessoa):
    def __init__(self, nome):
        Pessoa.__init__(self, nome)
        self.time = []

    def adicionar_pokemon(self, pokemon):
        self.time.append(pokemon)

    def remover_pokemon(self, pokemon):
        self.time.remove(pokemon)
```

```python
treinador = Treinador("Ash")
print(treinador.nome)
treinador.adicionar_pokemon("Pikachu")
treinador.adicionar_pokemon("Charmander")
treinador.adicionar_pokemon("Squirtle")
treinador.adicionar_pokemon("Bulbassaur")
print(treinador.time)
treinador.remover_pokemon("Squirtle")
print(treinador.time)
```

```
Ash
['Pikachu', 'Charmander', 'Squirtle', 'Bulbassaur']
['Pikachu', 'Charmander', 'Bulbassaur']
```

Resolução Exercício 3

FIAP

```python
class Treinador(Pessoa):
    def __init__(self, nome, time=[]):
        Pessoa.__init__(self, nome)
        self.time = time

    def adicionar_pokemon(self, pokemon):
        self.time.append(pokemon)

    def remover_pokemon(self, pokemon):
        self.time.remove(pokemon)

    def get_time(self):
        return self.time
```

```python
class Batalha:
    def __init__(self, treinador1, treinador2):
        self.treinador1 = treinador1
        self.treinador2 = treinador2

    def determinar_vencedor(self):
        time1 = self.treinador1.get_time()
        time2 = self.treinador2.get_time()

        soma_ataque1 = 0
        soma_ataque2 = 0
        for pokemon in time1:
            soma_ataque1 += pokemon.get_ataque()
        for pokemon in time2:
            soma_ataque2 += pokemon.get_ataque()

        if soma_ataque1 > soma_ataque2:
            return self.treinador1.get_nome()
        else:
            return self.treinador2.get_nome()
```

Resolução Exercício 4

FIAP

```python
class Pokebola:
    def __init__(self, pokemon):
        self.pokemon = pokemon

    def liberar(self):
        pokemon = self.pokemon
        self.pokemon = None
        return pokemon

    def trocar(self, novo_pokemon):
        pokemon_anterior = self.pokemon
        self.pokemon = novo_pokemon
        return pokemon_anterior
```

```python
pikachu = Pokemon("Pikachu", "Elétrico", 50, 40)
charmander = Pokemon("Charmander", "Fogo", 39, 50)

pokebola = Pokebola(pikachu)
print(pokebola.pokemon.nome) # "Pikachu"

troca = pokebola.trocar(charmander)
print(troca.nome) # "Pikachu"
print(pokebola.pokemon.nome) # "Charmander"

liberacao = pokebola.liberar()
print(liberacao.nome) # "Charmander"
print(pokebola.pokemon) # None
```

Resolução Exercício 5

```python
class LojaPokemon:
    def __init__(self):
        self.itens = {}

    def adicionar_item(self, nome, preco):
        self.itens[nome] = preco

    def remover_item(self, nome):
        del self.itens[nome]

    def realizar_compra(self, treinador, nome, quantidade):
        preco = self.itens[nome]
        custo = preco * quantidade
        if treinador.dinheiro >= custo:
            treinador.dinheiro -= custo
            treinador.itens_comprados[nome] = quantidade
            return True
        else:
            return False
```

```python
class Treinador:
    def __init__(self, nome, dinheiro):
        self.nome = nome
        self.dinheiro = dinheiro
        self.itens_comprados = {}

    def mostrar_itens_comprados(self):
        if self.itens_comprados:
            print(f"Itens comprados por {self.nome}:")
            for nome, quantidade in self.itens_comprados.items():
                print(f"- {nome}: {quantidade}")
        else:
            print(f"{self.nome} não possui itens comprados.")
```

```python
# Criando a loja
loja = LojaPokemon()

# Adicionando itens à loja
loja.adicionar_item("Pocao", 10)
loja.adicionar_item("Pokebola", 20)

# Criando um treinador
ash = Treinador("Ash", 100)

# Realizando compras
sucesso = loja.realizar_compra(ash, "Pocao", 2)
if sucesso:
    print("Compra realizada com sucesso!")
else:
    print("Saldo insuficiente.")

sucesso = loja.realizar_compra(ash, "Pokebola", 3)
if sucesso:
    print("Compra realizada com sucesso!")
else:
    print("Saldo insuficiente.")

# Mostrando itens comprados pelo treinador
ash.mostrar_itens_comprados()
```

```
Compra realizada com sucesso!
Compra realizada com sucesso!
Itens comprados por Ash:
- Pocao: 2
- Pokebola: 3
```

Resolução Exercício 6

```python
class Pokedex:
    def __init__(self):
        self.pokemons = {}

    def adicionar_pokemon(self, nome, tipo, ataque, defesa):
        self.pokemons[nome] = {
            "tipo": tipo,
            "ataque": ataque,
            "defesa": defesa
        }

    def pesquisar_pokemon(self, nome):
        if nome in self.pokemons:
            pokemon = self.pokemons[nome]
            return f"Nome: {nome}\nTipo: {pokemon['tipo']}\nAtaque: {pokemon['ataque']}\nDefesa: {pokemon['defesa']}"
        else:
            return "Pokemon não encontrado."
```

```python
pokedex = Pokedex()
pokedex.adicionar_pokemon("Pikachu", "Elétrico", 55, 40)
pokedex.adicionar_pokemon("Charizard", "Fogo/Voador", 84, 78)

print(pokedex.pesquisar_pokemon("Pikachu"))
print(pokedex.pesquisar_pokemon("Charizard"))
print(pokedex.pesquisar_pokemon("Squirtle"))
```

```
Nome: Pikachu
Tipo: Elétrico
Ataque: 55
Defesa: 40
Nome: Charizard
Tipo: Fogo/Voador
Ataque: 84
Defesa: 78
Pokemon não encontrado.
```

FIAP

This presentation has been designed using images from Flaticon.com
Images from Pokémon: The Pokémon Company