{introcomp}

Working 07: POO

Objetivos:

- Quanto ao conceito: Entender os conceitos básicos da Programação Orientada a Objetos e as suas implicações no desenvolvimento de softwares.
- Quanto a criação de classes: Saber modelar e implementar classes em python.

Praticando

1. Faça um programa que calcule a distância entre dois pontos. Para isso implemente a classe *Ponto*, cujos atributos privados deverão ser as coordenadas x e y. A classe deverá possuir não só o método construtor, mas também os métodos acessores (*get* e *set*) e um método que calcule a distância entre dois pontos.

   As entradas deverão ser quatro números reais represenando respectivamente as coordenadas x1 e y1 do primeiro ponto e as coordenadas x2 e y2 do segundo ponto.

   A saída deverá ser um número real, com duas casas decimais apenas, representando a distância entre os dois pontos.

Exemplo de Entrada	Exemplo de Saida
2 0 2 5.5	5.50
0 9 12 0	15.00

2. Utilizando a classe implementada no exercício anterior, crie a classe *Triangulo*, cujos atributos privados deverão ser três pontos distintos. Considere que os pontos não sejam colineares. Além de possuir os métodos acessores, a classe deverá também possuir um método que calcule o perímetro de um triângulo.

   As entradas deverão ser seis numeros reais, representando de dois em dois as coordenadas dos três vértices do triângulo.

   A saída deverá ser o perímetro do triângulo, represetada por um número real com duas casas decimais apenas.

Exemplo de Entrada	Exemplo de Saida
0 0 0 3 4 0	12.00
0 0 10 0 5 5	24.14

3. Implemente a classe *Carro* que possua os atributos velocidade media, consumo(dada em km/litro), capacidade do tanque de combustivel e quantidade atual de combustivel. Além disso, essa classe também deverá possuir alguns métodos:

   (a) O método Viajar, que recebe a quantidade de quilômetros como parâmetro.

   (b) O método Abastecer, que recebe a quantidade de combustivel a ser abastecida

   (c) O método Completar, que enche o tanque de combustivel até o limite da capacidade.

   Seu programa deve começar lendo os atributos do carro: velocidade média, consumo e capacidade do tanque, respectivamente. Considere que o carro começa com o tanque cheio.

   Após ler esses valores e criar um *Carro*, seu programa receberá comandos que serão associados com cada método de carro: Ao ler "Viaja", a próxima linha será um número real que representa a quantidade de km que serão percorridos, e portanto, deve-se chamar o método Viajar e usar a quantidade de km lida como argumento do método. Ao ler "Abastece", a próxima linha será um número real que representa quantos litros de combustivel devem ser abastecidos, e portanto, deve-se chamar o método Abastecer e usar a quantidade de litros lida como

argumento do método.Por fim, ao ler "Completa", seu programa deve chamar o método Completar, que não possui argumentos.

Seu programa deve receber comandos até que, ao invés de receber um dos três comandos listados acima, receba o comando "Encerra", que deve encerrar o programa.

As saídas possíveis são três:

(a) Ao chamar o método Viajar, deve-se imprimir "O carro andou KM km em H horas e gastou L litros de combustivel. O carro agora possui *Latual* litros de combustivel."substituindo os valores em maiúsculo pelos valores apropriados.

(b) Ao chamar o método Abastecer, deve-se imprimir "O carro foi abastecido com L litros. O tanque agora esta com *Latual* litros de combustivel."caso o tanque não tenha sido abastecido até sua capacidade máxima. Caso o tanque tenha sido abastecido até sua capacidade máxima, deve-se imprimir "O carro foi abastecido com L litros e esta com o tanque cheio!".

(c) Ao chamar o método Completar, também deve-se imprimir "O carro foi abastecido com L litros e esta com o tanque cheio!".

OBS: Considere que o o comando "Viaja"nunca enviará um número de KMs maior do que o máximo que o carro consegue percorrer com o combustivel atual. Além disso, ao receber o comando "Abastece"e receber um número de litros de combustivel que faça com que o tanque transborde, considere que o número passado foi exatamente o necessário para completar o tanque.

Exemplo de Entrada	Exemplo de Saida
60 12 100 Viaja 900 Abastece 40 Completa Encerra	O carro andou 900.00 km em 15.00 horas e gastou 75.00 litros de combustivel. O carro agora possui 25.00 litros de combustivel. O carro foi abastecido com 40.00 litros. O tanque agora esta com 65.00 litros de combustivel. O carro foi abastecido com 35.00 litros e esta com o tanque cheio!

4. Retorne ao exercício 2, seu dever agora é aprimorá-lo. No método construtor, você deverá verificar se os pontos dados realmente formam um triângulo. Caso os pontos sejam colineares, o programa deverá ser abortado infomando a mensagem "ERRO! Os pontos dados nao formam um triangulo",(Dica: é possível realizar essa verificação utilizando geometria analítica, pesquise!).

   Caso os pontos formem um triângulo, então o programa deverá informar a mensagem "Os pontos dados formam um triangulo <*tipo do triângulo*>"Ex: "Os pontos dados formam um triangulo equilatero". Os tipos são equilátero, isóceles ou escaleno.

   Logo após isso, o programa receberá UM número inteiro sendo 1 para calcular o perímetro ou 2 para calcular a área do triângulo. Considere que não haja entradas diferentes de 1 ou 2.

   Incialmemte o programa receberá seis números reais repesentando, duas as duas, as coordenadas dos vértices do triângulo. O programa fará a verificação de existência, e só após o programa receberá o número inteiro representando a opção que o usuário quer calcular.

Exemplo de Entrada	Exemplo de Saida
0 0 1 1 2 2 1	ERRO! Os pontos dados nao formam um triangulo
0 0 10 0 5 5 2	Os pontos dados formam um triangulo isoceles 25.00

Desafios

1. Pokepy!
   Para treinar suas habilidades com a programação orientada a objeto, vamos retormar a nossas infâncias e implementar uma batalha Pokémon! Você deve implementar duas classes: A classe *Pokemon* e a classe *ataque*.

   A classe *Pokemon* contém as informações do pokemon, que são: nome, pontos de vida, pontos de energia e ataques(sempre dois ataques apenas).

   Enquanto isso, a classe *Ataque* deve conter as seguintes informações: nome do ataque, pontos de dano que ele causa no oponente e pontos de energia que ele consome.

   Criadas as classes, seu objetivo agora é fazer com que a batalha pokemon aconteça!
   O primeiro passo é ler as informações dos pokemons que irão batalhar. Serão sempre dois, e a leitura deve ocorrer da seguinte forma:
   –> A primeira linha de entrada são as informações do primeiro pokemon (nome, vida e energia, nessa ordem);
   –> A segunda linha são as informações dos ataques desse pokemon (nome, dano e energia consumida, nessa ordem);

   Essa leitura deve acontecer para os dois pokemons que irão batalhar. Então, se quisermos realizar uma batalha entre **Pikachu** (com **100 pontos de vida** e **100 pontos de energia**, e ataques **ChoqueDoTrovao** que dá **20 pontos de dano** e consome **40 pontos de energia** e **InvestidaTrovao** que dá **10 pontos de dano** e consome **20 pontos de energia**) e **Squirtle** (com **120 pontos de vida** e **80 pontos de energia**, e ataques **Hidrobomba** que dá **25 pontos de dano** e consome **35 pontos de energia** e **GiroRapido** que dá **10 pontos de dano** e consome **30 pontos de energia**) devemos fornecer a seguinte entrada:

Exemplo de Entrada
Pikachu 100 100 ChoqueDoTrovao 20 40 InvestidaTrovao 10 20 Squirtle 120 80 Hidrobomba 25 35 GiroRapido 10 30

Escolher o primeiro pokemon: Devemos escolher ele aleatoriamente, importando a biblioteca random e usando a seed "pokemon" SEMPRE!;

Um turno de batalha: Um turno consiste em os dois pokemons se atacarem. O primeiro ataca o segundo e logo depois o segundo ataca o primeiro. No final de cada turno, os dois pokemons são restaurados 20 pontos de energia. O turno deve acabar se um dos pokemons for derrotato, ou seja, vida igual a zero (ela nunca fica menor que zero). Nesse caso, o jogo acaba e o vencedor é o pokemon vivo;

O ataque entre os pokemons: No ataque, um pokemon ataca o outro com um de seus dois golpes (que será escolhido aleatoriamente). Com esse golpe, o pokemon atacante deve ser descontado a energia que o ataque consome, e o oponente descontado o dano que o ataque causa. Mas atenção! Se o ataque que o pokemon vai dar consome mais energia que ele tem, ele deve usar seu outro golpe. Caso ele ainda não tenha energia para usar a alternativa, ele não faz nada!

A saída esperada: Para cada turno, seu programa deve descrever algumas coisas:
–> Quem atacou e o nome do ataque;
–> Quanta vida e energia cada pokemon tem depois desse ataque. Isso deve ocorrer duas vezes por turno, pois são dois ataques. Por exemplo, veja como fica a saída do primeiro turno da batalha proposta acima:

Perceba que, para apenas um turno, imprimimos na tela duas interações: A do ataque do primeiro pokemon (Choque de trovao do Pikachu) e a do ataque do segundo pokemon (HIdrobomba do Squirtle). No final, você deve falar quem ganhou. Veja o exemplo do último turno da batalha acima:

Exemplo de Saída
Pikachu usa ChoqueDoTrovao. Pikachu: 100.00 de vida 80.00 de energia Squirtle: 100.00 de vida 80.00 de energia Squirtle usa Hidrobomba. Pikachu: 75.00 de vida 80.00 de energia Squirtle: 100.00 de vida 65.00 de energia

Exemplo de Saída
Pikachu usa Investida$_t rovao$. Pikachu: 5.00 de vida 20.00 de energia Squirtle: 10.00 de vida 35.00 de energia Squirtle usa Hidrobomba. Pikachu: 0.00 de vida 20.00 de energia Squirtle: 10.00 de vida 20.00 de energia O vencedor foi Squirtle!!!