INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS PAU DOS FERROS
BR 405, KM 154, Chico Cajá

**Professor:** Demetrios Coutinho
**Curso:** TADS
**Disciplina:** POO

Implemente o seguinte Sistema com atributos privados e seus métodos públicos. Para cada atributo privado faça seus gets e sets.

1) Classe **Person**:
    a. **Atributos**: name, age
    b. **Métodos**:
        i. Construtor(name,age)
        ii. __str__
    c. Uma Pessoa pode ter qualquer nome, mas a idade precisa ser um valor entre 1 e 150 (inclusive). Se uma idade inválida for fornecida, inclua uma mensagem de erro informando que a idade precisa estar no intervalo [1, 150].
    d. O método (__str__) deve retornar uma string no seguinte formato: {name}({age}), onde {name} e {age} representam os valores de campo correspondentes. Ex: Chloe(24)
2) Classe **Hobby**
    a. Herda de Enum
    b. **Atributos**: MUSIC=1,SPORTS=2,GAMES=3
3) Classe **Friend**:
    a. **Herda** Person
    b. **Atributos**: hobby
    c. **Métodos**:
        i. Construtor(name,age,hobby)
        ii. chill ()
        iii. play(friends)
        iv. __str__
    d. O método **chill** deve retornar uma string do seguinte formato {name} is chilling Ex: Ben is chilling.
    e. O método **play** retorna uma string que depende do número de amigos no array que é passado como argumento.
        1. No caso de uma lista vazia (sem amigo):jogar {hobby};
        2. No caso de um amigo (lista com um elemento):jogar {hobby} com {friend};
        3. No caso de dois amigos (lista com dois amigos):jogar {hobby} com {friend1} e {friend2}
        4. Se a lista de argumentos contiver mais de dois amigos: jogando {hobby} com {friend1}, {friend2},.. . , e {friendN} Observe que os amigos são separados por vírgula e espaço mais a palavra e antes do último amigo.

Em todos os casos, {hobby} deve ser substituído pelo valor do campo correspondente, e os amigos individuais devem ser representados por seus nomes. Por exemplo: tocando música; jogando com Alex; tocando música com Chris e Chloe; praticando esportes com Dan, Lisa e Ben.

O método (__str__) deve retornar a mesma string da classe Person mais um espaço e o hobby : {name}({age}) {hobby}Ex: Chloe(24) GAMES

4) **Interface** Nuisance
    a. **Método**: annoy()
5) Classe **Telemarketing**
    a. Implementa a interface Nuisance
    b. **Herda** de Person
    c. **Métodos**:
        i. giveSalesPitch()
    d. O método **giveSalesPitch** deve retornar uma string do seguinte formato {name} pressiona os outros a comprar coisas. Ex: Chap pressiona os outros a comprar coisas.
    e. O método **annoy** deve retornar uma string do seguinte formato {name} irrita ao dar um discurso de vendas Ex: Chap irrita ao dar um discurso de vendas.
6) Classe **Mosquito**
    a. **Implementa** Nuisance
    b. **Herda** Insect
    c. **Métodos**: buzz
    d. O método **buzz** deve retornar a string no seguinte formato: {species} buzzing around. Ex: Culex tarsalis buzzing around
    e. O método **annoy** deve retornar a string: “buzz buzz buzz”
7) Classe **Insect**
    a. **Atributos**: species
    b. **Métodos**:
        i. Construtor(species)
        ii. __str__
    c. O método toString (__str__) deve retornar uma string no seguinte formato: {className}: {species}Ex: Insect: wasp
8) Classe **Butterfly**
    a. **Herda** Insect
    b. **Atributos:** Colors
    c. **Métodos:**
        i. Construtor(species,colors,butterfly=None)
        ii. __str__
    d. No construtor quando os campos species e butterfly forem fornecidos, você deve inicializar o objeto com esses valores. Caso, um objeto Butterfly for fornecido você deve inicializar com base nesse objeto.
    e. O método **toString** (__str__) deve retornar uma string no seguinte formato: {species}[{color1},{color2},. . ,{colorN}]{color1}, {color2}, {colorN}] são substituídos pelas cores individuais na lista. Se nenhuma cor for fornecida, os

colchetes vazios serão adicionados à string. Ex: Monarch [amarelo, laranja, preto]
Ex:Morpho[azul]; Ex: Phoebis []

9) **Sobrecarregando o método __eq__:**
Determine onde sobrescrever o método __eq__ para fornecer o seguinte comportamento desejado: Ser capaz de determinar se dois amigos são iguais em tempo de execução. Em outras palavras: quando criar dois objetos **Friend**, e ambos os amigos têm o mesmo nome, a mesma idade e o mesmo hobby, então esses dois amigos devem ser iguais. Caso contrário, eles não devem ser iguais.

**No fim, faça um main.py testando todos os métodos.**