SCC0504 - Programação Orientada a Objetos

Classes e Objetos: Uso, Vida e Memória.

Prof.: Leonardo Tórtoro Pereira

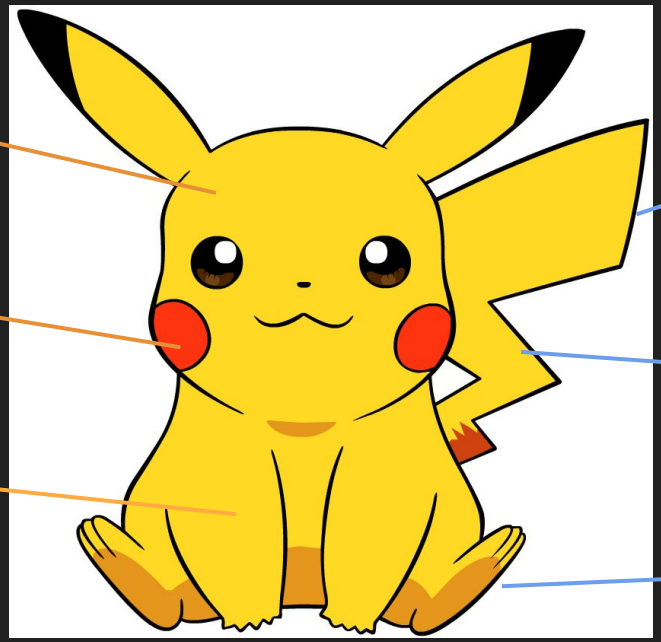
leonardop@usp.br

Objeto = Características + Comportamento

Nome:
Pikachu

Tipo:
Elétrico

Cor:
Amarelo



Thunder()

TailWhip()

Run()

Objeto

Objeto = Características + Comportamento

normalmente são representadas como variáveis

```
skills = ["fire blast",
     "scratch"];

 sAttack = 300;


 hp = 280;


 status = "healthy"
```

Habilidades

Ataque Especial

HP

Estado

Os valores podem se alterar: poison, receber dano, aprender habilidade nova, paralisado.

Objeto

Objeto = Características + Comportamento

normalmente são representadas como funções

Poison

```
poison(target){
     target.sufferDamage(
          target.getHP() * 0.05);
     target.setStatus("poisoned");
}
```

Synthesis

```
synthesis(){
     self.recoverHP(self.maxHP*0.5);
}
```

Altera as variáveis de seus atributos e também de outros objetos!

Classe

- Alguns objetos podem ser agrupados em um mesmo tipo, pois possuem características e comportamentos em comum.
- Classes servem como um molde para a criação de objetos similares.

Classe

```
public class Pokémon{
    ???
}
```

Classe

```
public class Pokémon{
  int hp, atk, def, speed;
  int level, xp;
  String name, type;
  void takeDamage(int amount)
  {
       hp = hp - amount;
  }
  void getXP(int amount)
  {
       level++;
  }
}
```

Classe

```
class PokemonWorld{
    ???
}
```

Classe

```
class PokemonWorld{
  public static void main(){
    Pokemon mimikyu;
    mimikyu = new Pokémon();
    mimikyu.getExp(20);
  }
}
```

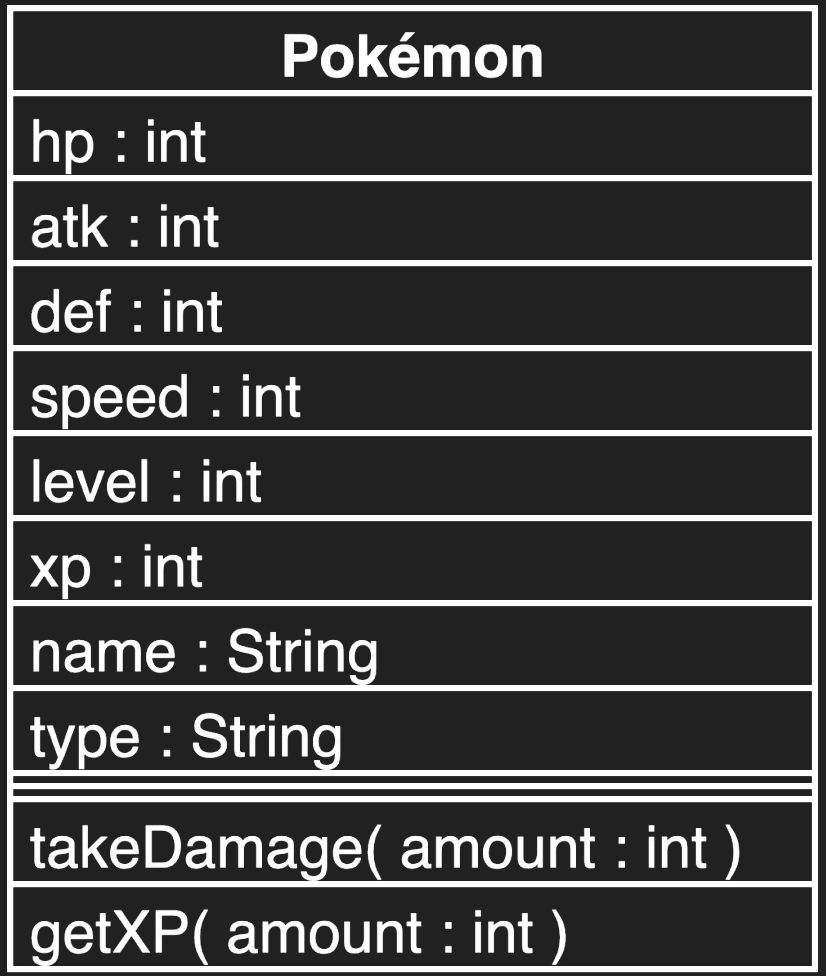

Pokémon
hp : int
atk : int
def : int
speed : int
level : int
xp : int
name : String
type : String
takeDamage(amount : int)
getXP(amount : int)

Diagrama de Classes

Diagrama de Classes

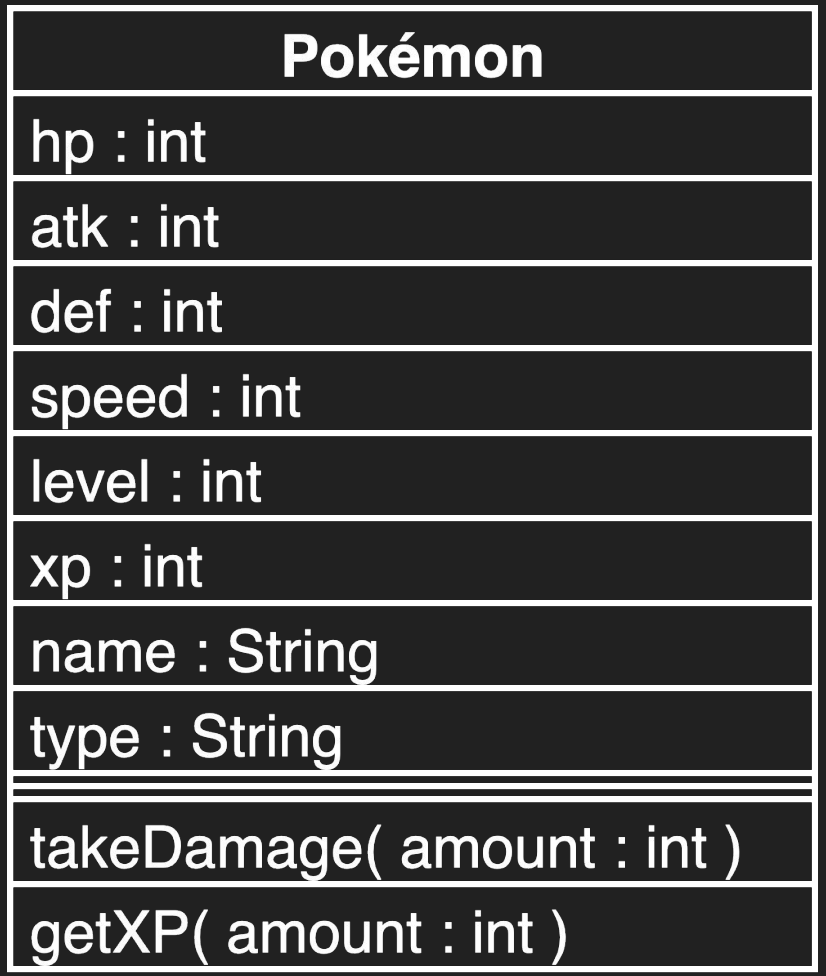
- É um diagrama estrutural que descreve a estrutura de um sistema ao mostrar suas classes e seus atributos, operações (métodos) e relações entre objetos
- É o principal componente da modelagem orientada a objetos
- As classes no diagrama representam os principais elementos, interações na aplicação e classes a serem programadas

Diagrama de Classes

- As classes são caixas com 3 compartimentos
  - O do topo é o nome da classe
    - Em negrito (alguns programas colocam em itálico)
    - Primeira letra maiúscula
    - Centralizado
  - O do meio são os atributos
    - Alinhados à esquerda
    - Primeira letra minúscula

Diagrama de Classes

- As classes são caixas com 3 compartimentos
  - O de baixo são as operações (métodos)
    - Alinhados à esquerda
    - Primeira letra minúscula


Pokémon
hp : int
atk : int
def : int
speed : int
level : int
xp : int
name : String
type : String
takeDamage(amount : int)
getXP(amount : int)

Visibilidade

Visibilidade

- Determina “quem” pode acessar os membros (atributo e método) de uma classe
- Public
  - Qualquer um pode acessar. Independente de classe, pacote, etc.
- Private
  - Apenas objetos da classe que contém aquele membro podem acessá-lo

Visibilidade

- ➔ Existem outros níveis de visibilidade
- ➔ Alguns variam de acordo com especificidades da linguagem
- ➔ Inclusive, caso você não coloque nenhum, cada linguagem assumirá um valor *default*
- ➔ Vamos vê-los no tempo certo :)

Visibilidade

```
public class Pokémon{
  private int hp, atk, def, speed;
  private int level, xp;
  private String name, type;
  public void takeDamage(int amount)
  {
       hp = hp - amount;
  }
  public void getXP(int amount)
  {
       level++;
  }
}
```

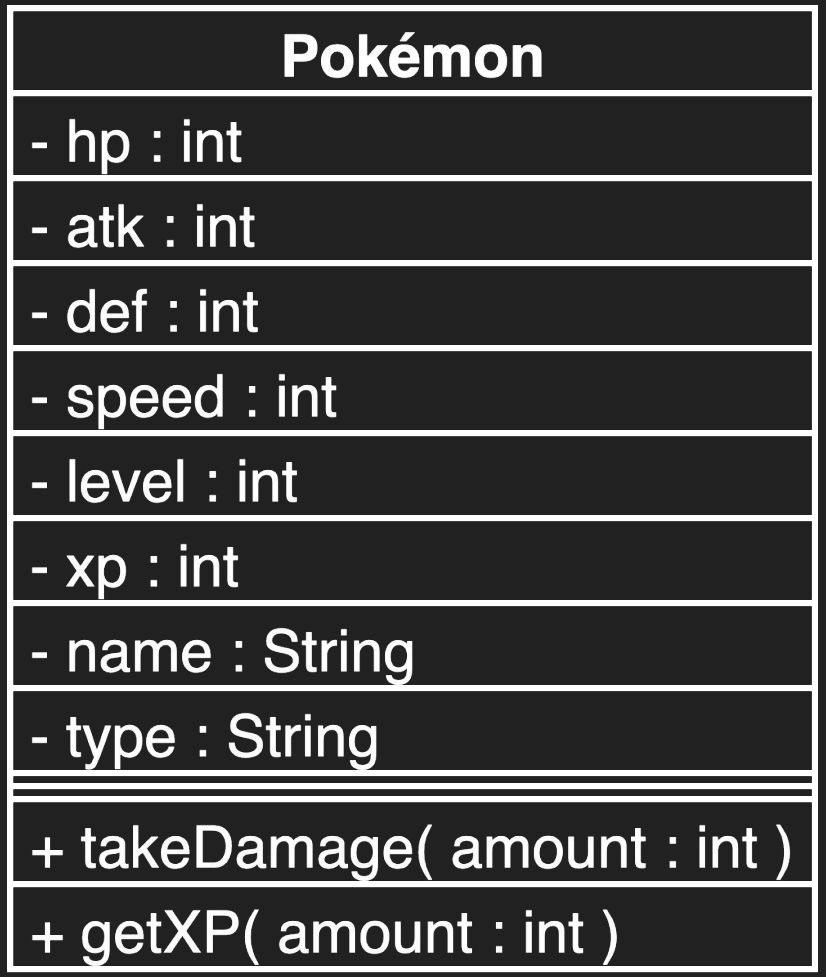

Pokémon
- hp : int
- atk : int
- def : int
- speed : int
- level : int
- xp : int
- name : String
- type : String
+ takeDamage(amount : int)
+ getXP(amount : int)

Diagrama de Classes

- Visibilidade dos membros (atributo e método)
  - Colocadas antes do nome do membro
  - + -> Public
  - - -> Private
  - # -> Protected
  - ~ -> Package

Exemplos em Outras Linguagens (C#)

```
public class Pokémon{
  private int hp, atk, def, speed;
  private int level, xp;
  private String name, type;
  public void takeDamage(int amount)
  {
       hp = hp - amount;
  }
  public void getXP(int amount)
  {
       level++;
  }
}
```

Exemplos em Outras Linguagens (C#)

```
class PokemonWorld{
  static void Main(){
    Pokemon mimikyu;
    mimikyu = new Pokémon();
    mimikyu.getExp(20);
  }
}
```

Exemplos em Outras Linguagens (C++)

```
class Pokémon{
    private:
        int hp, atk, def, speed;
        int level, xp;
        String name, type;
    public:
        void takeDamage(int amount){
            hp = hp - amount;
        }
        void getXP(int amount){
            level++;
        }
}
```

Exemplos em Outras Linguagens (C++)

```
int main(void){
    Pokemon mimikyu;
    mimikyu = new Pokémon;
     mimikyu.getExp(20);
    Pokemon gengar;
    gengar.getExp(30);  //Alocação na pilha
  }
}
```

Getter e Setter

Getter e Setter

```
public class Pokémon{
  private int hp, atk, def, speed;
  private int level, xp;
  private String name, type;
  public void takeDamage(int amount)
  {
       hp = hp - amount;
  }
  public void getXP(int amount)
  {
       level++;
  } ...
```

Getter e Setter

```
public class Pokémon{
...
    public int getHP(){
        return hp;
    }
    public void setHP(int _hp){
        hp = _hp;
    }
    public void setAtk(int atk){
        this.atk = atk;
    }
}
```

Construtor

Construtor

- Ao instanciar uma nova classe, um método *construtor* é chamado
- Ele é usado para inicializar as variáveis de sua classe
  - E até mesmo chamar outros métodos na instanciação
- Você não precisa definir um *construtor*
  - Nesse caso, ele será vazio
- Normalmente, sua sintaxe é um método com o nome da classe

Construtor

```
public class Pokémon{
  private int hp, atk, def, speed;
  private int level, xp;
  private String name, type;
  public void takeDamage(int amount)
  {
       hp = hp - amount;
  }
  public void getXP(int amount)
  {
       level++;
  }
}
```

Construtor

```
public class Pokémon{
    public Pokemon(){
    }
}
```

Construtor

```
public class Pokémon{
    public Pokemon(int _level, int _xp, String _name){
        name = _name;
        level = _level;
        xp = _xp;
    }
}
```

Construtor

```
class PokemonWorld{
  public static void main(){
    Pokemon mimikyu;
    mimikyu = new Pokémon(1, 0, “Definitely Pikachu”);
    mimikyu.getExp(20);
  }
}
```

Destrutor (C++)

Exemplos em Outras Linguagens (C++)

```
int main(void){
    Pokemon mimikyu;
    mimikyu = new Pokémon();
     mimikyu.getExp(20);
    Pokemon gengar;
    gengar.getExp(30);  //Alocação na pilha
    delete mimikyu;
  }
}
```

Referências

- https://en.wikipedia.org/wiki/Class_diagram
- https://www.uml-diagrams.org/class-diagrams-overview.html