

---

# Programação Orientada a Objetos

## Conceitos Básicos de Orientação a Objetos

---

# Objetivos Instrucionais

---

O aprendiz é capaz de definir responsabilidades de classes óbvias

O aprendiz é capaz de definir colaborações de responsabilidades

O aprendiz é capaz de definir e aplicar os conceitos de classe e método

O aprendiz é capaz de definir e aplicar os conceitos de mensagem, instância e inicialização

---



**CLASSE:** Pessoa, Animal,  
Coisa, Conceito ou Evento  
relevante ao sistema  
em questão

**INSTÂNCIA / OBJETO**

Rex Bidu ...



**CLASSE:** Pessoa, Animal, Coisa, Conceito ou Evento relevante ao sistema em questão

**INSTÂNCIA / OBJETO**

João José ...

# Atividade 3

---

- Definir as ações que um cachorro faz ou sabe
  - Definir as ações que um carteiro faz ou sabe
  - Tempo: 5m
  - Em grupo de 3
-

# Uma Solução para a Atividade 3

---

**Cachorro protege casa**

**Cachorro late**

**Cachorro ataca carteiro**

**...**

**Carteiro entrega cartas**

**Carteiro foge de cachorros**

**•  
•  
•**

# Atividade 4

---

- Definir as ações que uma pessoa faz ou sabe
  - Definir as ações que um ponto gráfico faz ou sabe
  - Tempo: 5m
  - Em grupo de 3
-

# Uma Solução para a Atividade 4

---

- Definir as ações que uma pessoa faz ou sabe
    - Sabe o nome
    - Sabe a idade
    - Pensa
    - Anda
    - Dança
-



# Uma Solução para a Atividade 4

---

- Definir as ações que um ponto gráfico faz ou sabe
    - Sabe sua posição (coordenadas)
    - Sabe sua cor
    - Move-se
    - Compara-se a outro ponto
-

# RESPONSABILIDADE

---

■ Constitui o que a classe **faz** ou **sabe**:

- O que um objeto faz = ação que ele realiza!
- O que um objeto sabe = conhecimento que ele mantém

# Atividade 5

---

- **Identifique as responsabilidades das classes Cachorro e Carteiro**
  - **Identifique as responsabilidades das classes Pessoa e Ponto Gráfico**
  - **Tempo: 05m**
  - **Em grupo de 3**
-

# Uma Solução da Atividade 5

---

Cachorro protege casa

Cachorro late

Cachorro ataca carteiro

...

Carteiro entrega cartas

Carteiro foge de cachorros

- 
- 
- 

Cachorro

Proteger casa

Latir

Atacar carteiro

Entregar cartas

...

Carteiro

Entregar cartas

Fugir de cachorros

...

# Uma solução da Atividade 5

---

Pessoa sabe o nome

Pessoa sabe a idade

Pessoa pensa

Pessoa anda

...

Ponto Gráfico Sabe posição

Ponto Gráfico sabe cor

Ponto Gráfico move-se

Ponto Gráfico compara-se

...

Pessoa

Sabe o nome

Sabe a idade

Pensa

...

Ponto Gráfico

Sabe posição

Sabe cor

Move-se

...

# COLABORAÇÃO

---

- Para realizar a lógica de uma dada responsabilidade, um objeto da classe pode:

- Realizar a tarefa sozinha

ou

- Solicitar a colaboração de objetos de outra(s) classe(s) servidora(s)!
-

# Atividade 6

---

- Uma colaboração corresponde efetivamente a que?
  - Exemplifique uma colaboração que uma responsabilidade da classe Carteiro precise!
  - Tempo: 3m - Individual
-

# Uma Solução para a Atividade 6

---

- Uma colaboração de uma responsabilidade de classe cliente corresponde a uma responsabilidade de classe servidora
  - A responsabilidade “Entregar Cartas” da classe Carteiro pode fazer uso da responsabilidade “Entregar Cartas” da classe Cachorro!
-



# Atividade 7 – Para o Lar

---

- ❑ Suponha que voce queira enviar flores para sua avó Rosa, que mora em Lorena, e voce está impossibilitado de entregar pessoalmente
  - ❑ Suponha, então, que voce vá à uma florista, chamada Maria, e encomende a entrega das flores à sua avó
  - ❑ Descreva as pessoas ou entidades envolvidas nessa tarefa, em termos de responsabilidades, colaborações e associações entre as pessoas e entidades
-

# Atividade 8 – Para o Lar

---

- Descreva os conceitos de objeto, classe, instância, mensagem, método, ocultamento da informação (information hiding), polimorfismo e herança com relação aos entes da Atividade 7
  - No caso de herança, estabeleça uma hierarquia que envolva os entes da Atividade 7
  - Neste último caso, caracterize os conceitos de subclasse e superclasse, classes concretas e classes abstratas!
-

# Atividade 9 – Para o Lar

---

- Um contador digital é um contador limitado, que volta para o valor mínimo quando ele alcança o valor máximo. Exemplos incluem os números em um relógio digital e o hodômetro de um carro
  - Defina uma descrição de classe para um contador limitado. Considere as seguintes responsabilidades: definir valores mínimo e máximo, iniciar contador, parar contador, incrementar o contador e retornar o valor corrente
  - Implemente em Python
-

# Atividade 10 – Para o Lar

---

- Dada a Atividade 9, considere o uso de uma classe dígito. Quais seriam as responsabilidades da classe dígito?
  - Reformule a solução do contador limitado, considerando o uso das classes contador limitado e dígito
  - Implemente em Python
-

---

# Classes e Métodos

---

# Atividade 11

---

Relembrando que **responsabilidade** é o que a classe **faz** ou **sabe**,

Definir as responsabilidades da classe `PlayingCard`, que representa uma carta de jogos de cartas

Grupo: 3 alunos

Tempo: 5 min

---

# Uma Solução para a Atividade 11

---

- Return suit and rank values
- Return color of face
- Draw card on playing surface
- Maintain face-up or face-down status

# Atividade 13

---

Com base nas responsabilidades definidas para a classe `PlayingCard`, propor os métodos correspondentes a elas.

Obs.: Uma responsabilidade pode dar origem a um ou mais métodos!

Grupo: 3 alunos

Tempo: 5min

---



# Uma Solução para a Atividade 3

---

- `color()` – return color value
  - `draw(x,y)` – draw card image
  - `flip()` – flip card image: face up or down
  - `getSuitValue()` – return suit value
  - `getRankValue()` – return rank value
-

# Atividade 14

---

Definir os métodos de acesso da classe PlayingCard

Obs.: Existem dois tipos de métodos de acesso de variáveis de instância: accessor (do tipo get ou getter) e mutator (do tipo set ou setter)!

Grupo: individual

Tempo: 5min

---

# Uma Solução para a Atividade 14

---

- `isFaceUp()` – test face-up or down
  - `setFaceUpDown(upDown)` – define face-up or down
  - `getSuitValue()` – return suit value
  - `setSuitValue(suitValue)` – define suit value
  - `getRankValue()` – return rank value
  - `setRankValue(rankValue)` – define rank value
-

# Criação de Novas Variáveis

---

- Como espaço é criado para novas variáveis?
    - Alocação Estática (Stack-based memory):
      - Através de declarações de variáveis → gerência de memória automática → gerência de memória baseada em pilha → pilha de execução
    - Alocação Dinâmica (Heap-based memory):
      - Através de diretivas explícitas → a gerência de memória pode não ser automática → gerência de memória baseada em heap
-

# Criação de Objetos em Python

---

```
aCard = PlayingCard( DIAMOND, 5);
```

# Métodos de Acesso

---

- O acesso a variáveis de instância deve ser mediado por métodos especiais definidos na classe.
  - Método *getter* ou de acesso propriamente dito: retorna simplesmente o valor da variável
  - Método *setter* ou mutator: atribui um novo valor à variável
-

# Criação e Inicialização

---

- Python fornece uma característica que amarra os aspectos de criação e inicialização dos objetos:  
**Construtores**
  - Isto evita os dois problemas seguintes:
    - Criar um objeto, mas não inicializar apropriadamente as variáveis de instância do objeto
    - Criar um objeto e inicializar as variáveis de instância mais de uma vez
  - Além dos construtores, certas linguagens usam os métodos de classe ou de fábrica
-

# Construtores

---

- Realiza as tarefas de alocação e inicialização das variáveis de instância dos objetos
  - Em Python um Construtor é um método com o nome `__init__`, implicitamente público, mas sem tipo de retorno
  - Pode ter lista de argumentos, como qualquer método
  - Um construtor é automaticamente invocado quando um objeto é criado.
-



# Exemplo de Construtor

---

```
class PlayingCard:
```

```
    # constructor
```

```
    def __init__(self, suitValue, rankValue):
```

```
        # inicialização
```

---

---

**FIM**

---