



Colégio
Pedro II

LINGUAGEM DE PROGRAMAÇÃO II

{ Programação O.O. (Classes e Objetos)
Professor: João Luiz Lagôas

}

Introdução à P.O.O.



- A orientação à objetos é um **paradigma** que representa toda uma filosofia para construção de sistemas.
- Utiliza uma ótica mais próxima do mundo real.
- É a forma de programação mais atual no Mercado.



Como era bem antes: Programação de Baixo Nível



Colégio
Pedro II

- Programação dedicada ao nível de hardware.
- Dependência com as máquinas.
- Código escrito em binário ou em micro instruções (linguagem de máquina).

```

1 FDX 12:01a 23- 1
A 002000 C2 30 REP #$30
A 002002 18 CLC
A 002003 F8 SED
A 002004 A9 34 12 LDA #$1234
A 002007 69 21 43 ADC #$4321
A 00200A 8F 03 7F 01 STA $017F03
A 00200E D8 CLD
A 00200F E2 30 SEP #$30
A 002011 00 BRK
A 2012

r
PB PC NUmxDIZC .A .X .Y SP DP DB
; 00 E012 00110000 0000 0000 0002 CFFF 0000 00
g 2000

BREAK

PB PC NUmxDIZC .A .X .Y SP DP DB
; 00 2013 00110000 5555 0000 0002 CFFF 0000 00
m 7f03 7f03
>007F03 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00:UU.....
  
```

Como era antes:

Programação Estruturada



- Paradigma de programação baseado fortemente na modularização, cuja idéia é dividir o programa em unidades menores: **funções** ou **procedimentos**.



Como era antes: Programação Estruturada



Colégio
Pedro II

```
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
# define MAX 100 /* constant*/

    struct addr { /*struct called list*/
        char name[30] ;
        char street[40] ;
        char town[20] ;
        char county[20] ;
        char code[10] ;
    } list[MAX]; /* 100 records*/

main()
{
    int choice;
    init_list(); /* initialize the list */
    for(;;) {
        choice = menu_select(); /* get user's selection*/
        switch(choice) {
            case 1: enter(); /* enter a new entry */
                break;
            case 2: del(); /* delete an entry */
                break;
            case 3: show_list(); /* display the list */
                break;
            case 4: search(); /* find an entry */
                break;
            case 5: save(); /* save to disk */
                break;
            case 6: load(); /* read from disk */
                break;
            case 7: exit(0);
        }
    }
}
```

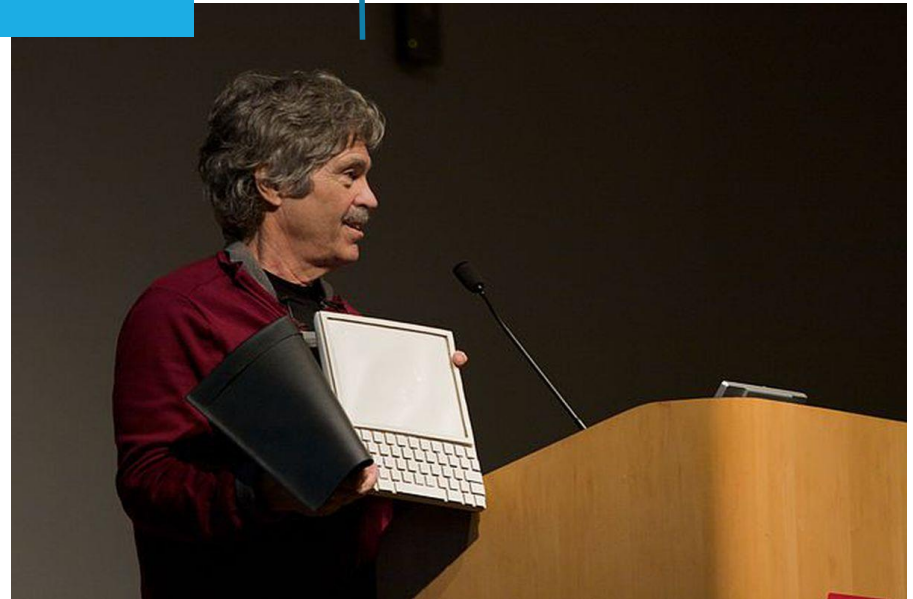


Mudança de Paradigma



“O computador ideal deve funcionar como um organismo vivo, isto é, cada célula se relaciona com outras afim de alcançar um objetivo, mas cada uma funciona independentemente. As células poderiam também reagrupar-se para resolver um outro problema ou desempenhar outras funções.”

Alan Kay
Pai da Programação O.O.



Mudança de Paradigma

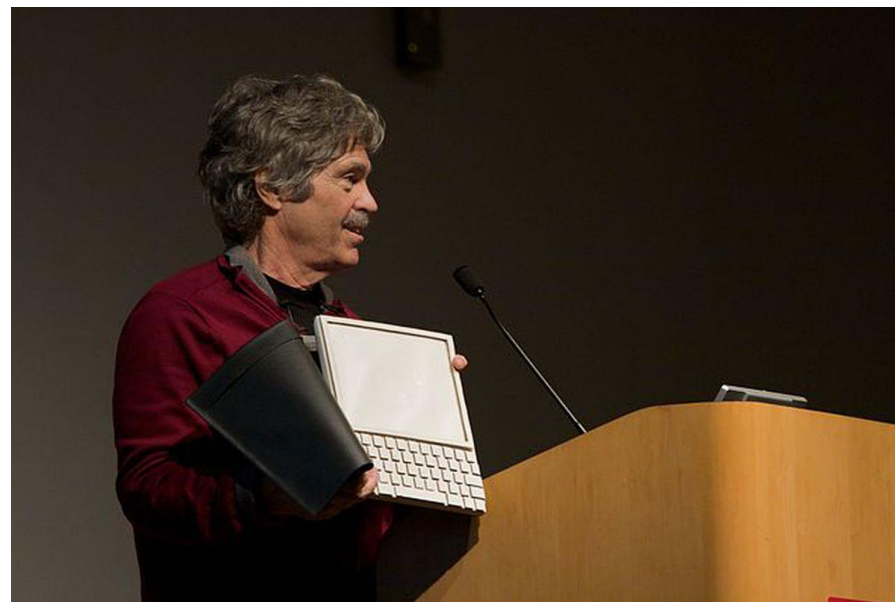


Primeira Linguagem de Programação que seguia o Paradigma de Programação Orientado a Objetos.



Alan Kay

Pai da Programação O.O.



Como é hoje: Programação O.O.



- Paradigma baseado nos conceitos de **objetos**.



O que é um objeto?



- **Definição de Dicionário:** um **objeto** é toda coisa material que pode ser percebida pelos sentidos.
- **Definição de P.O.O.:** Um **objeto** é toda coisa material ou abstrata que pode ser percebida pelos sentidos e descrita por meio das suas **características** e **comportamentos** além de ter um **estado atual** definido.

O que é um objeto?



- Mesa
- Caneta
- Copo
- Relógio
- Compromisso
- Aluno
- Bolo
- Data
- Tempo
- Janela
- Botão
- Pokémon
- Personagem
- Livro

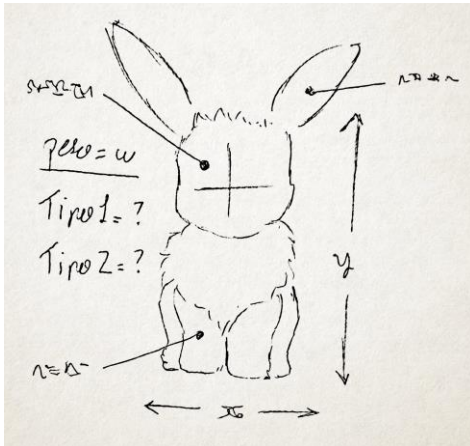


Analizando um objeto

Pokémon



- **Características (o que o objeto tem):**



- **Comportamentos (o que o objeto faz):**

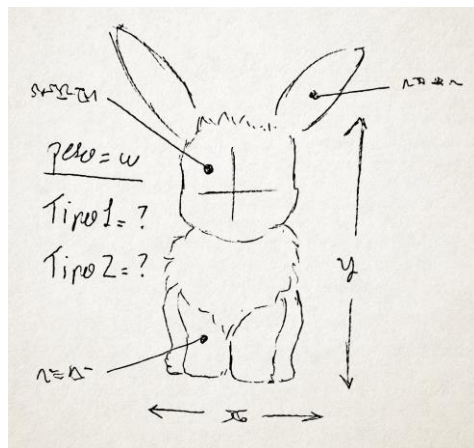
Analizando um objeto

Pokémon



- **Características (o que o objeto tem):**
 - Nome:
 - Tipo:
 - Número:

- **Comportamentos (o que o objeto faz):**
 - Atacar
 - Defender
 - Esquivar



Analizando um objeto

Pokémon



- **Características (o que o objeto tem):**
 - Nome: **Pikachu**
 - Tipo: **Elétrico**
 - Número: **025**

} Estado Atual
- **Comportamentos (o que o objeto faz):**
 - **Atacar**
 - **Defender**
 - **Esquivar**

Analizando um objeto

Pokémon



- **Características (o que o objeto tem):**

- Nome: Charizard
- Tipo: Fogo
- Número: 006

} Estado Atual

- **Comportamentos (o que o objeto faz):**

- Atacar
- Defender
- Esquivar

Analizando um objeto

Pokémon



- **Características (o que o objeto tem):**
 - Nome: Jigglypuff
 - Tipo: Normal
 - Número: 039

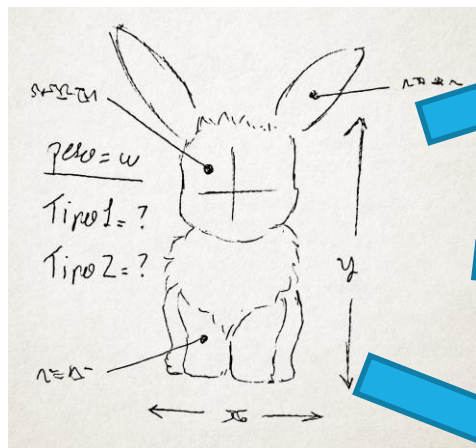
} Estado Atual
- **Comportamentos (o que o objeto faz):**
 - Atacar
 - Defender
 - Esquivar

Analizando um objeto

Pokémon



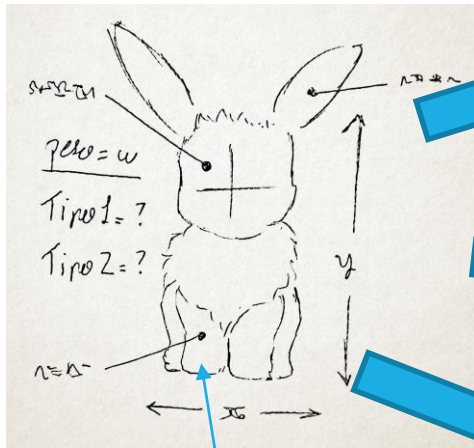
Colégio
Pedro II



Analizando um objeto Pokémon



Colégio
Pedro II



classe



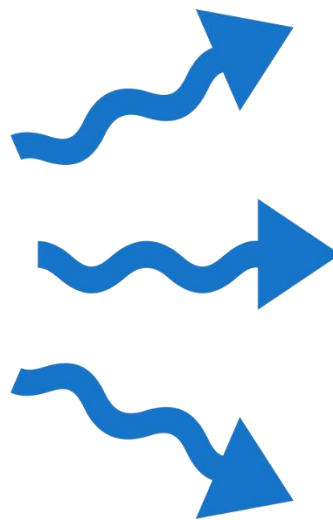
objetos

Analizando um objeto

Outro Exemplo: Bolo



Colégio
Pedro II

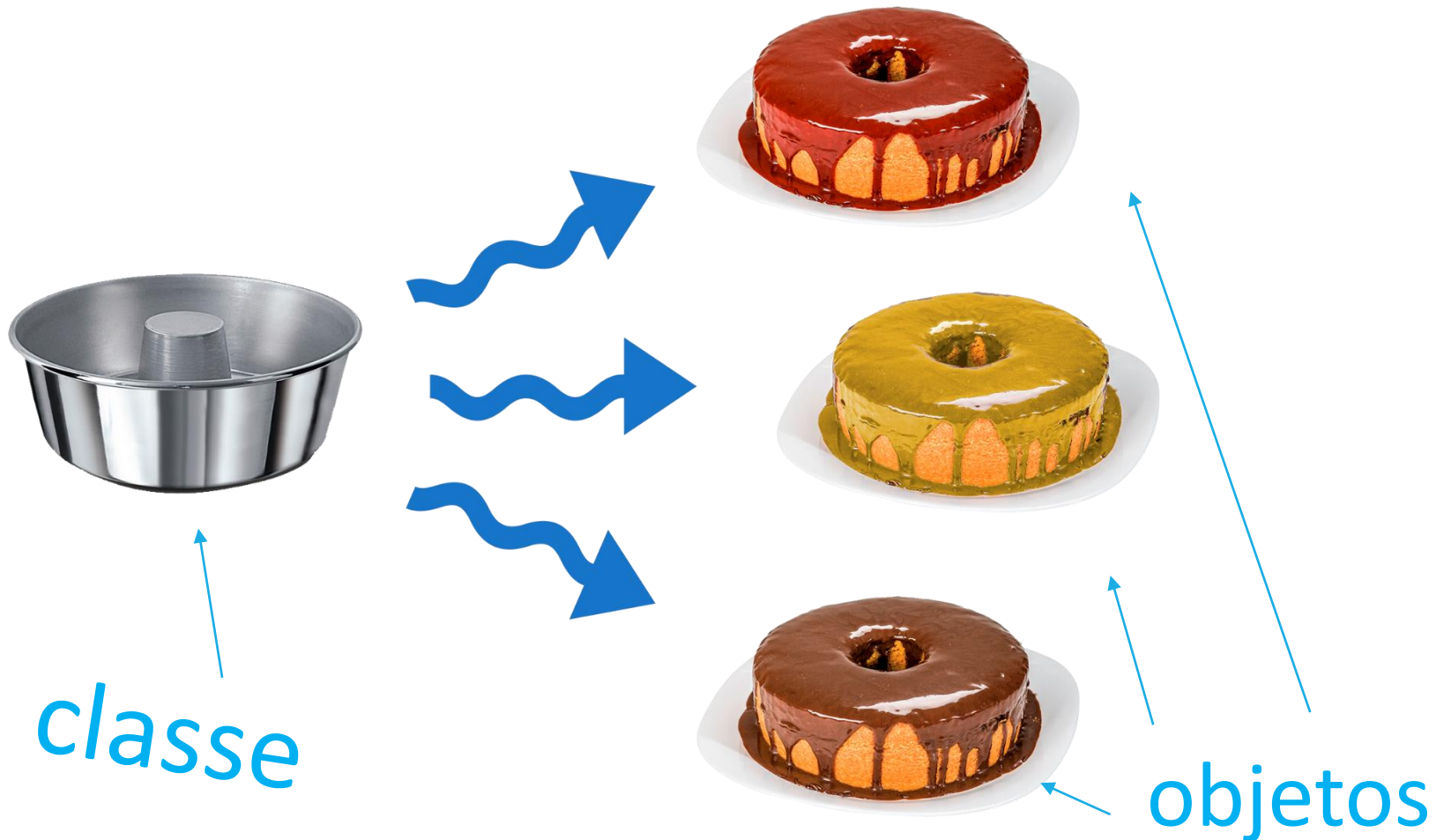


Analizando um objeto

Outro Exemplo: Bolo



Colégio
Pedro II





Classe

- A **classe** é uma estrutura estática utilizada para descrever objetos mediante atributos e métodos (funcionalidades).
- A classe é um modelo ou template para a criação de objetos.
- Em outras palavras, a classe determina as **características** e os **comportamentos** de uma coleção de objetos.

Exemplos de classes:

- Entidades do negócio da aplicação (pessoa, conta, cliente, fornecedor);
- Entidades de interface (janela, botão, painel, frame, barra);
- Entidades abstratas de tecnologia (conexão com banco de dados, um arquivo-texto, um serviço web).

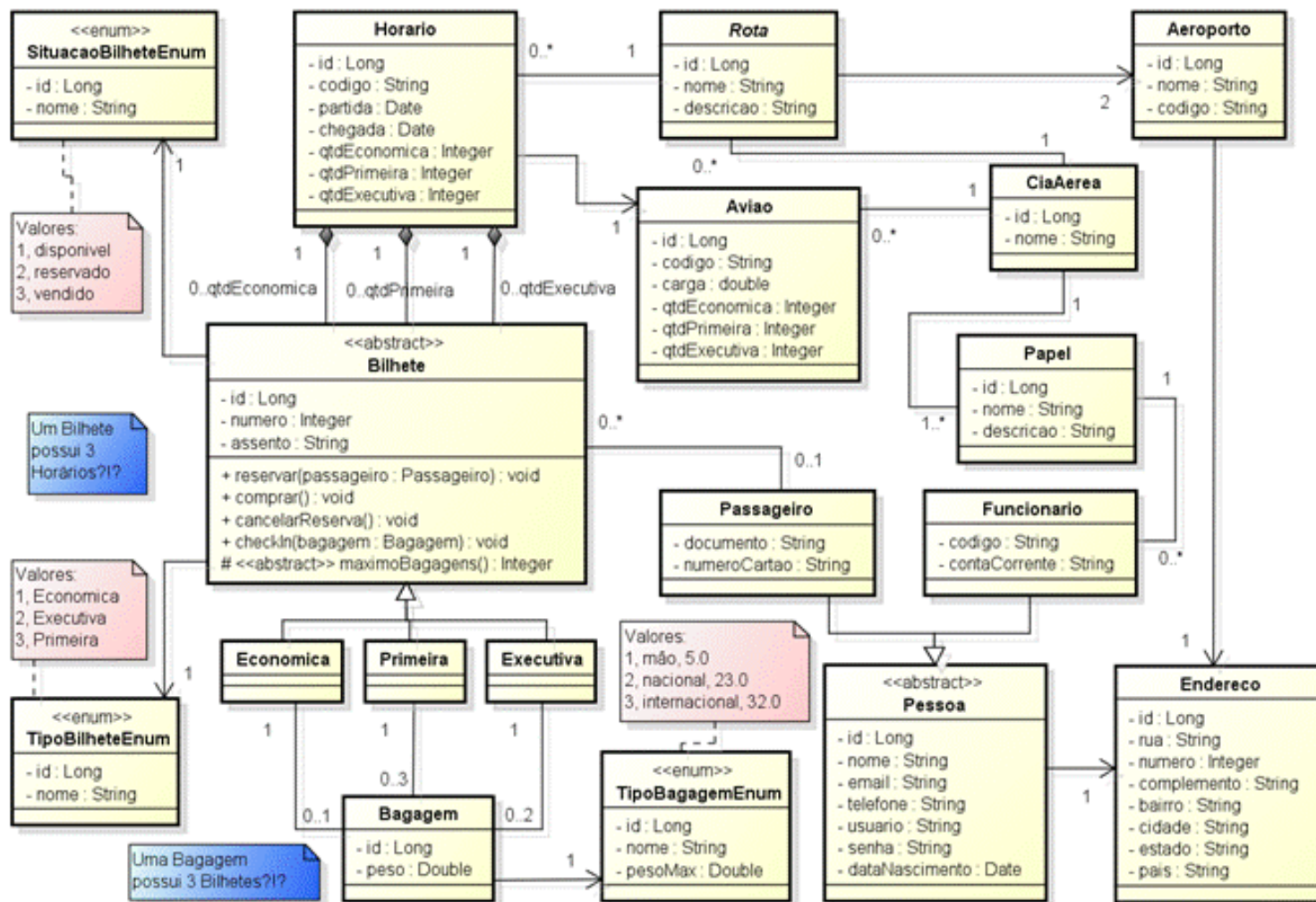


Objeto

- Um **objeto** é uma estrutura dinâmica originada com base em uma classe.
- Após utilizar uma classe para criar objetos dizemos que estamos **instanciando** objetos.
- Em outras palavras, um objeto é uma instância de uma classe com um **estado atual** definido.
- Para instanciar um objeto de uma determinada classe, procedemos para a declaração de uma variável qualquer e lhe atribuímos o operador *new* seguido do nome da classe (seu **construtor**) que desejamos instanciar.



Por que usar P.O.O.?





Por que usar P.O.O.?

- **Confiabilidade**
- **Manutenível**
- **Extensível**
- **Reuso**
- **Natural / Abstração**



Exemplo Prático

Classe



Colégio
Pedro II

```
class Relogio
{
    public int Horas;
    public int Minutos;
    public int Segundos;

    public void MostrarHorario()
    {
        Console.WriteLine("{0}:{1}:{2}",
            Horas, Minutos, Segundos);
    }
}
```



Exemplo Prático

Classe



```
class Relogio
```

Atributos

```
{  
    public int Horas;  
    public int Minutos;  
    public int Segundos;  
}
```

```
public void MostrarHorario()  
{
```

```
    Console.WriteLine("{0}:{1}:{2}",  
        Horas, Minutos, Segundos);  
}
```

```
}
```

Métodos



Exemplo Prático

Objeto



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();

        rel.Horas = 4;
        rel.Minutos = 57;
        rel.Segundos = 39;

        rel.MostrarHorario();

        Console.ReadLine();
    }
}
```

Exemplo Prático

Objeto



Colégio
Pedro II

rel



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();

        rel.Horas = 4;
        rel.Minutos = 57;
        rel.Segundos = 39;

        rel.MostrarHorario();

        Console.ReadLine();
    }
}
```

Exemplo Prático

Objeto



Colégio
Pedro II

rel



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();

        rel.Horas = 4;
        rel.Minutos = 57;
        rel.Segundos = 39;

        rel.MostrarHorario();

        Console.ReadLine();
    }
}
```

```
public void MostrarHorario()
{
    Console.WriteLine("{0}:{1}:{2}",
        Horas, Minutos, Segundos);
}
```

4:57:39

Console

Exemplo Prático

Objeto



rel



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();

        rel.Horas = 4;
        rel.Minutos = 57;
        rel.Segundos = 39;

        rel.MostrarHorario();

        Console.ReadLine();
    }
}
```

Aguardando o usuário digitar algo para finalizar o programa

Criando mais objetos da mesma classe



- Um outro ponto interessante quando programamos utilizando classes, é o **reaproveitamento de código**.
- A partir de uma mesma classe, podemos criar vários objetos que se comportam de modo semelhante mas apresentam características particulares.
- Basta instanciarmos vários objetos de uma mesma classe!

Exemplo Prático

Objeto



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();
        Relogio rel2 = new Relogio();

        rel.Horas = 4;
        rel.Minutos = 57;
        rel.Segundos = 45;

        rel2.Horas = 12;
        rel2.Minutos = 30;
        rel2.Segundos = 15;

        rel.MostrarHorario();

        rel2.MostrarHorario();

        Console.ReadLine();
    }
}
```

Exemplo Prático

Objeto



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();
        Relogio rel2 = new Relogio();

        rel.Horas = 4;
        rel.Minutos = 57;
        rel.Segundos = 45;

        rel2.Horas = 12;
        rel2.Minutos = 30;
        rel2.Segundos = 15;

        rel.MostrarHorario();

        rel2.MostrarHorario();

        Console.ReadLine();
    }
}
```


Exemplo Prático

Objeto



Colégio
Pedro II



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();
        Relogio rel2 = new Relogio();

        rel.Horas = 4;
        rel.Minutos = 57;
        rel.Segundos = 45;

        rel2.Horas = 12;
        rel2.Minutos = 30;
        rel2.Segundos = 15;

        rel.MostrarHorario();

        rel2.MostrarHorario();

        Console.ReadLine();
    }
}
```

Exemplo Prático

Objeto



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();
        Relogio rel2 = new Relogio();

        rel.Horas = 4;
        rel.Minutos = 57;
        rel.Segundos = 45;

        rel2.Horas = 12;
        rel2.Minutos = 30;
        rel2.Segundos = 15;

        rel.MostrarHorario();

        rel2.MostrarHorario();

        Console.ReadLine();
    }
}
```

Exemplo Prático Objetos



Nome do objeto:

rel

Valor dos atributos:

Horas = 4

Minutos = 57

Segundos = 45

Nome do objeto:

rel2

Valor dos atributos:

Horas = 12

Minutos = 30

Segundos = 15

objetos
na
memória

Exemplo Prático

Objeto



Colégio
Pedro II



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();
        Relogio rel2 = new Relogio();

        rel.Horas = 4;
        rel.Minutos = 57;
        rel.Segundos = 45;

        rel2.Horas = 12;
        rel2.Minutos = 30;
        rel2.Segundos = 15;

        rel.MostrarHorario();
```

```
rel2.MostrarHorario();
Console.WriteLine("{0}:{1}:{2}", rel.Horas, rel.Minutos, rel.Segundos);
}
```

4:57:39

Console

Exemplo Prático

Objeto



Colégio
Pedro II



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();
        Relogio rel2 = new Relogio();

        rel.Minutos = 30;
        rel2.Segundos = 15;

        rel.MostrarHorario();
        rel2.MostrarHorario();

        Console.ReadLine();
    }
}
```

```
public void MostrarHorario()
{
    Console.WriteLine("{0}:{1}:{2}",
        this.Horas, this.Minutos, this.Segundos);
}
```

12:30:15

Console

Exemplo Prático

Objeto



```
class Program
{
    static void Main(string[] args)
    {
        Relogio rel = new Relogio();
        Relogio rel2 = new Relogio();

        rel.Horas = 4;
        rel.Minutos = 57;
        rel.Segundos = 45;

        rel2.Horas = 12;
        rel2.Minutos = 30;
        rel2.Segundos = 15;

        rel.MostrarHorario();

        rel2.MostrarHorario();

        Console.ReadLine();
    }
}
```

Aguardando o usuário digitar algo para finalizar o programa

Prática



Adicione à classe Relogio o método TotalizarSegundos(). Esse método deverá calcular a quantidade de segundos total de acordo com os valores armazenados em seus atributos e retornar esse valor.

Após implementar o método, teste sua funcionalidade no método Main() através de uma instância de Relogio.

Sua assinatura deve ser: `public int TotalizarSegundos()`

```
class Relogio
{
    public int Horas;
    public int Minutos;
    public int Segundos;

    public void MostrarHorario()
    {
        Console.WriteLine("{0}:{1}:{2}", Horas, Minutos, Segundos);
    }

    public int TotalizarSegundos()
    {
        ?????????
    }
}
```

