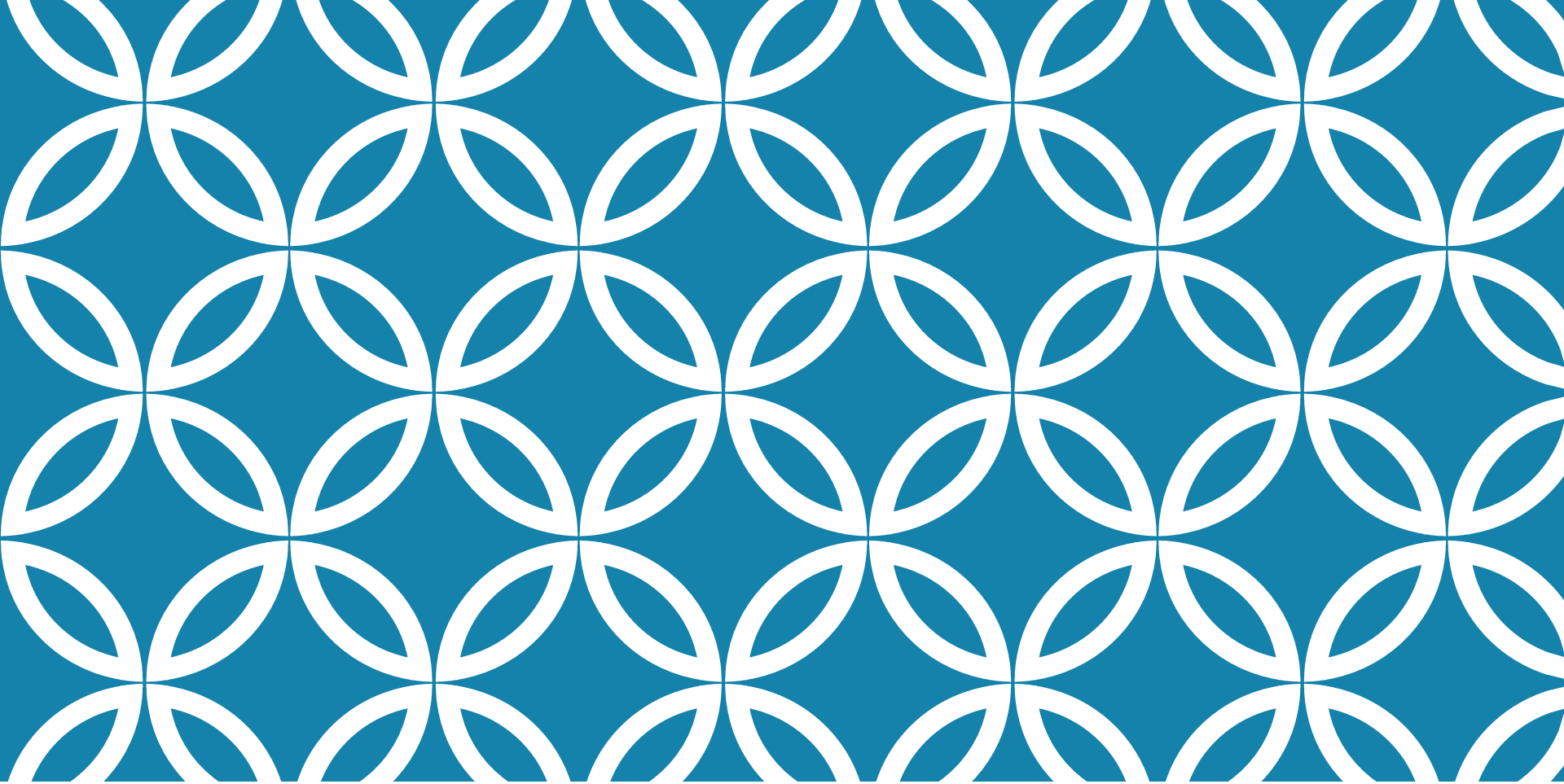


Orientação a Objetos

Rafael Vieira Coelho

Orientação a Objetos

Um **modelo de programação** ou
paradigma de programação

"Paradigma é um conjunto de regras que estabelecem fronteiras e descrevem como resolver os problemas dentro dessas fronteiras. Os paradigmas influenciam nossa percepção; ajudam-nos a organizar e a coordenar a maneira como olhamos para o mundo..."

Daniel C. Morris e Joel S. Brandon,
Reengenharia: reestruturando sua empresa,
São Paulo, Makron Books, 1994.

Orientação a Objetos

- É o modelo de programação mais adotado no desenvolvimento de sistemas corporativos
- Fácil de fazer manutenção das aplicações
- Diminuem a complexidade do desenvolvimento de sistemas

Orientação a Objetos

1950 – 1960 Era do Caos	1970 – 1980 Era da Estruturação	1990 até agora Era dos Objetos
Saltos, gotos, variáveis não estruturadas, variáveis espalhadas ao longo do programa	If-then-else Blocos Registros Laços-While	Objetos Mensagens Métodos Herança

O conceito de Orientação a Objetos data do final da década de 60 e início da década de 70

- Simula 67 (60's);
- Smalltalk (70's);
- C++ (80's).

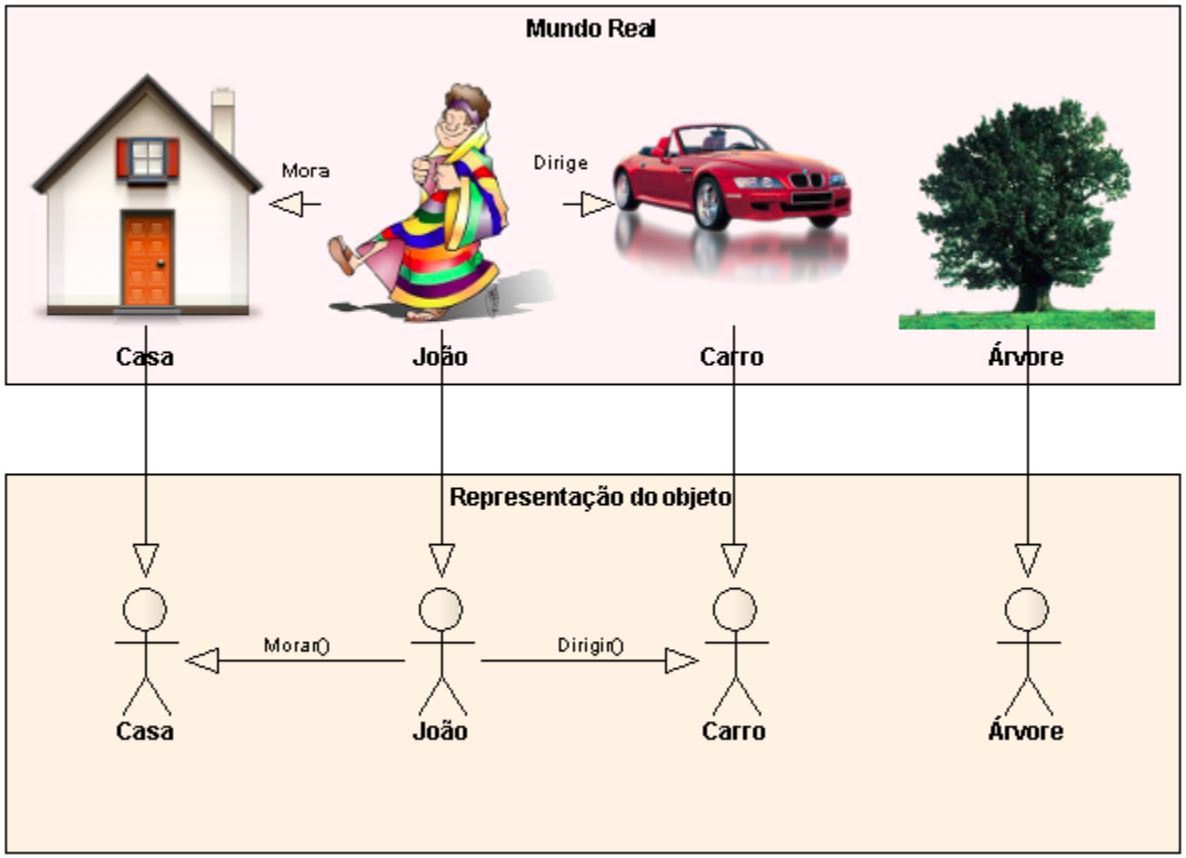
Surgiu a necessidade de modelar sistemas mais complexos.

O que Diabos são Objetos?

Objetos são a chave para entender a OO;

Se olharmos em nossa volta, encontraremos vários exemplos de objetos reais:

Celular;
Mesa;
Computador;
Janela;
Lâmpada;
Etc.



Objetos x Classes

- Quando queremos representar um objeto do mundo real com suas características e ações, precisamos definir um tipo novo em um novo arquivo (Classe) com suas variáveis e métodos próprios.

```
Public class Casa {
..
}
```

- Para criar um novo objeto desta classe de objetos (modelo), precisamos usar a palavra reservada **new**.

```
Casa x = new Casa();
```

Objetos x Classes

- Quando declaramos um objeto, mas não instanciamos um novo objeto para armazenar (new), o padrão é **null**

```
Casa x;
```

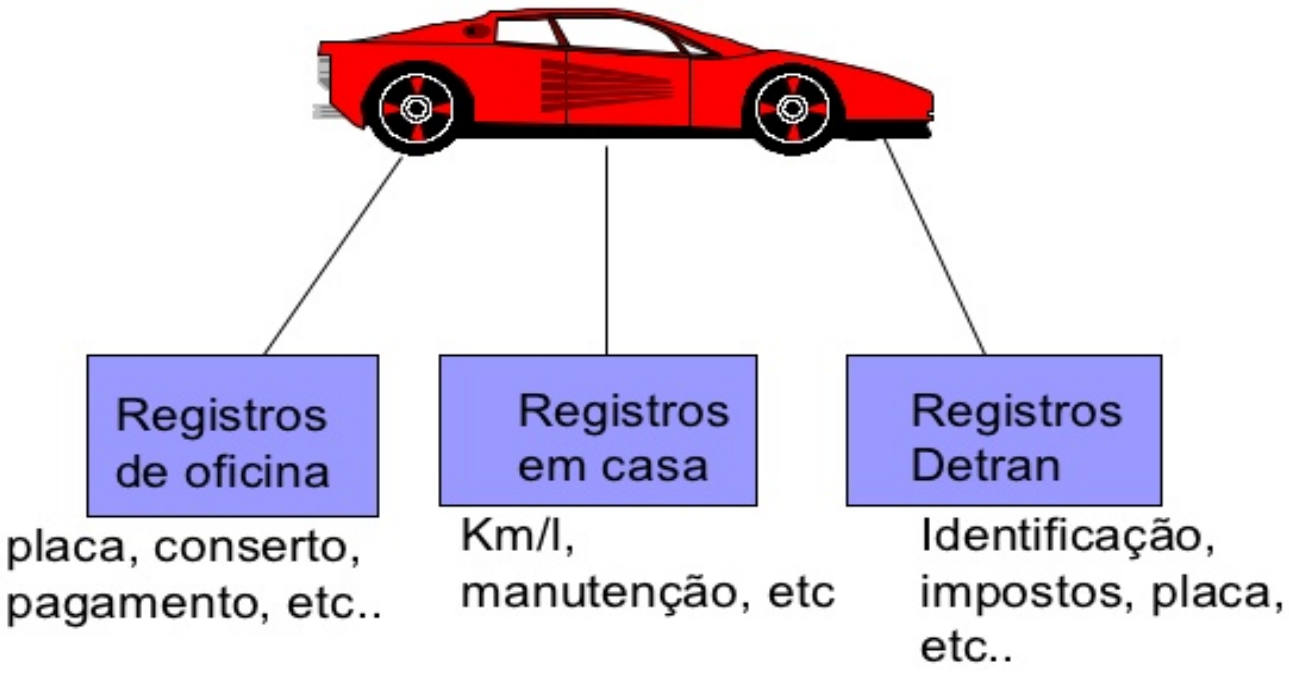
```
Casa x = null;
```

- Quando tentamos acessar um objeto não instanciado a exceção **NullPointerException** ocorre.

Como se Define uma Classe?

abstrair

Extrair de um conteúdo o que dele possa ter de mais interessante.



Definição de uma Classe de Objetos

Os objetos reais possuem duas características:

1) Objetos armazenam seu estado em **atributos:**
   * Correspondentes às variáveis em programação estruturada.

2) Objetos expõem seu comportamento através de **métodos:**
   * Correspondentes às funções em programação estruturada.

Por exemplo,
Um **aluno** pode ter:

1) nome, nota1, nota2, nota3, exame, …
2) calcularMedia(), alterarNota(),…

Atributos de uma Classe

Estado:

```
public class Student {
    private String name;
    private int ID;
    private double GPA;

...
```

Métodos de uma Classe

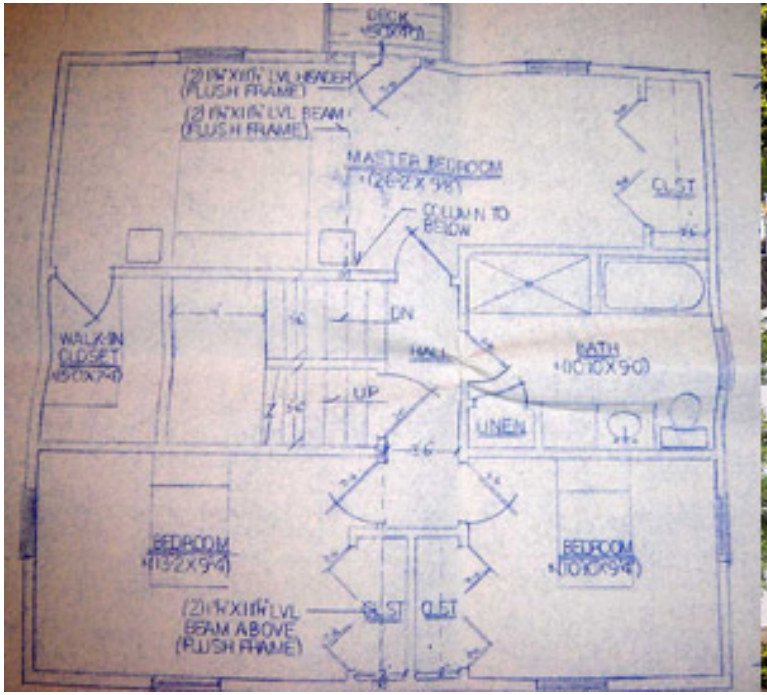
Estado:

```
public class Student {
   private String name;
   private int ID;
   private double GPA;

...
```

Comportamento:

```
...
   public boolean canGraduate() {
      return (GPA >= 2.0 && abs >= 5);
   }
}
```

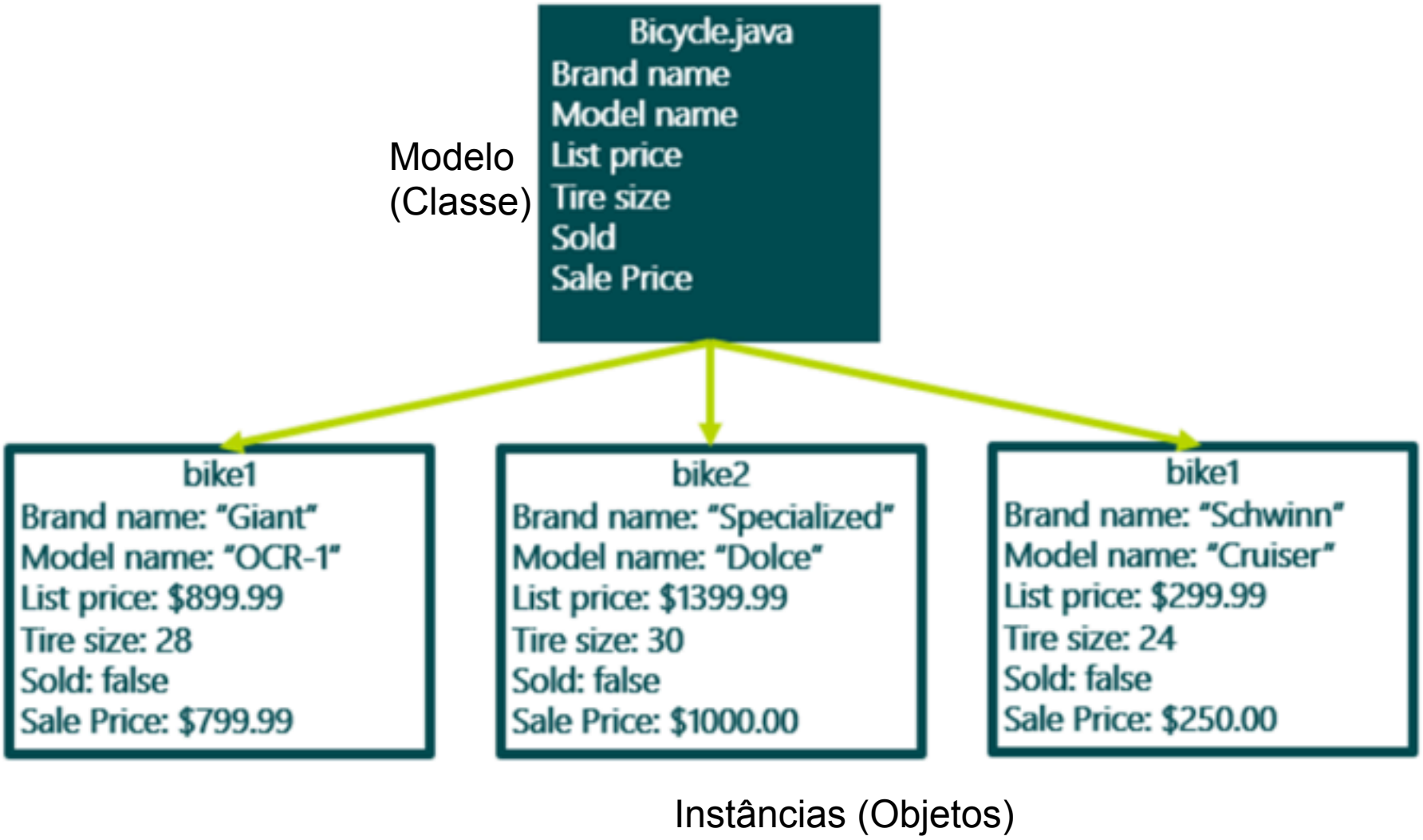
Classe x Objeto



Modelo

Instâncias

Classe x Objetos



Classe x Objetos

Modelo (Classe)

House Blueprint
State:
- Color
- Sq Footage
- Backyard
- Balcony

Behavior:
- Construct floor plan
- Paint Order
- Materials Order

Instâncias (Objetos)

House #1
State:
- blue
- 2000
- no
- no

Behavior:
- Construct floor plan
- Paint Order
- Materials Order

House #2
State:
- beige
- 7000
- no
- yes

Behavior:
- Construct floor plan
- Paint Order
- Materials Order

House #3
State:
- yellow
- 20000
- yes
- no

Behavior:
- Construct floor plan
- Paint Order
- Materials Order

Programação Orientada a Objetos

Empacotar o código em objetos individuais fornece:

1. Modularidade
   Objetos são independentes.

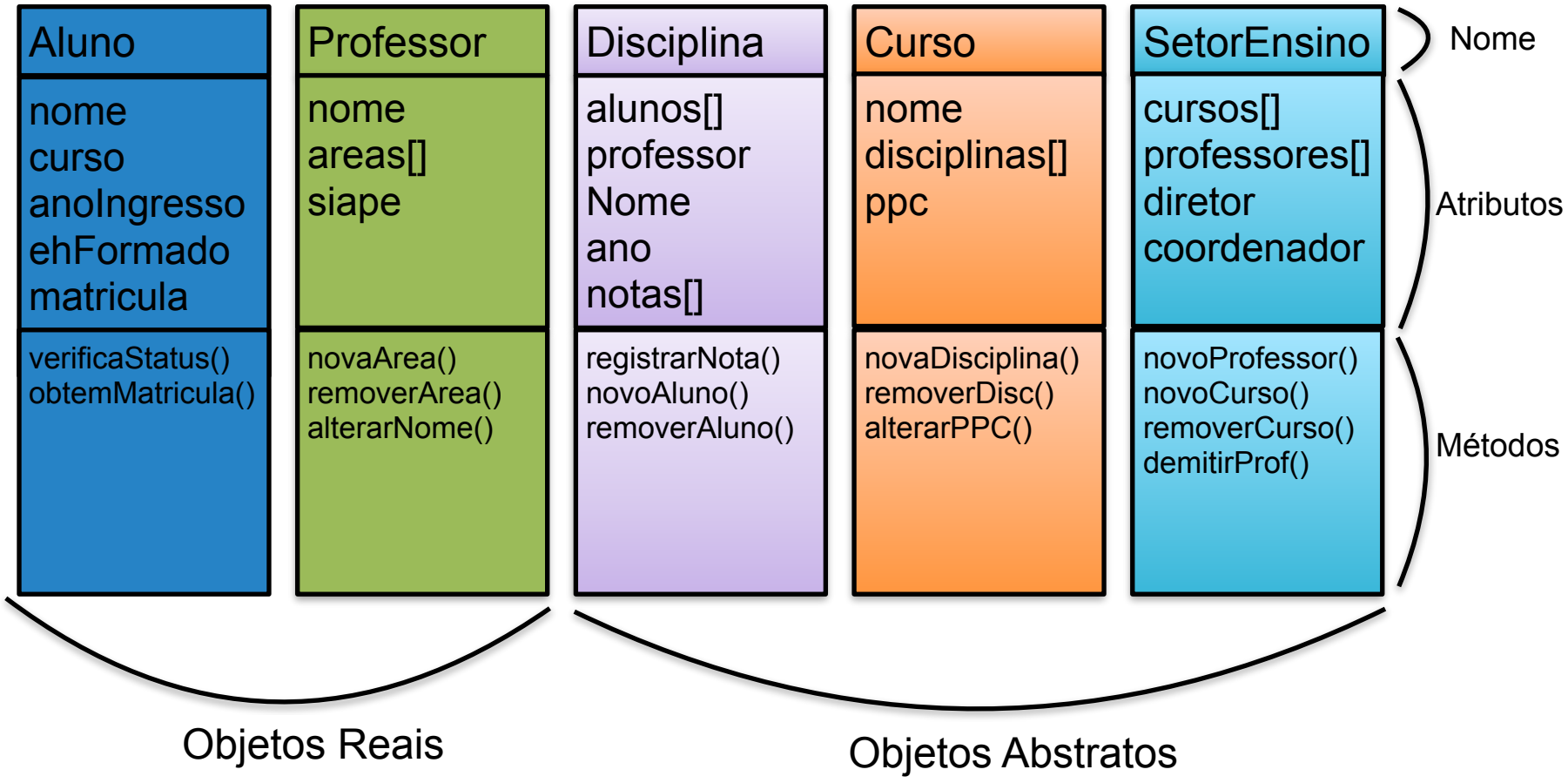
2. Ocultação de informação
   Os detalhes da implementação de um objeto permanecem ocultos.

3. Reuso
   Objetos podem ser reutilizados em diferentes programas.

4. Plugabilidade
   Objetos podem ser substituídos em um programa, como peças.

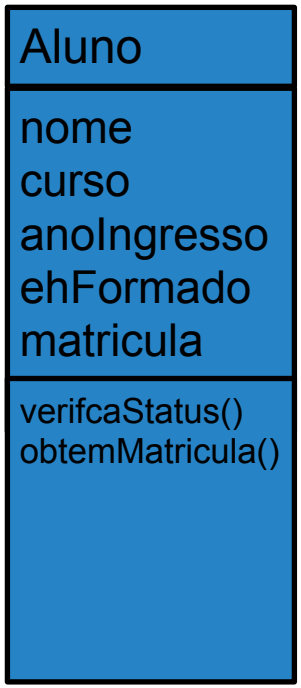
E se fôssemos criar um Sistema Acadêmico para o IFRS?

Quais classes teríamos em nosso *software*?



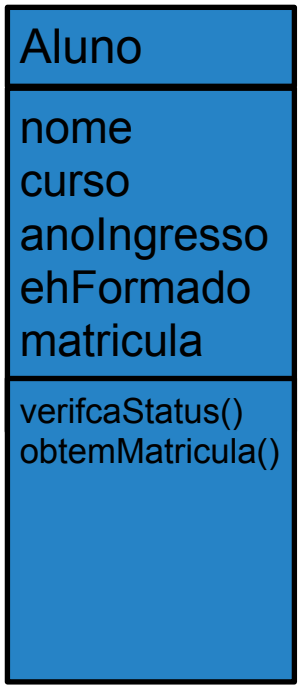
Classe Aluno

```
public class Aluno {
    //Atributos
    String nome;
    String curso;
    int anoIngresso;
    boolean ehFormado;
    long matricula;
    //Métodos
    String verificaStatus () {
        if (ehFormado) {
            return "O aluno ainda não completou os créditos";
        }
        return "O aluno entrou no ano" + anoIngresso +  "e se formou";
    }
    long obtemMatricula() {
        return matricula;
    }
}
```



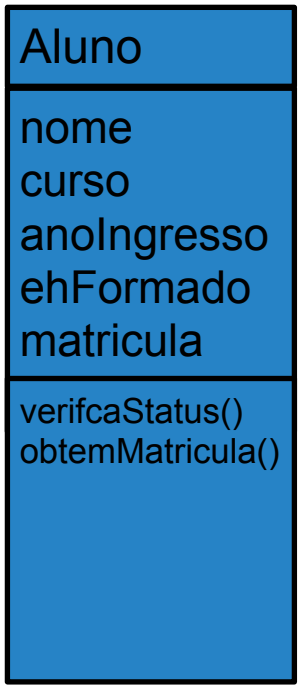
Como se cria um objeto Jorge da classe Aluno?

```
public class ProgramaTeste {
    public static void main(String []args) {
        Aluno jorge;

        jorge = new Aluno();
        jorge.nome = “Jorge Adão de Albuquerque”;
        jorge.curso = “Técnico em Informática Integrado ao Ensino Médio”;
        jorge.anoIngresso = 2019;
        jorge.ehFormado = false;
        jorge.matricula = 164090;
    }
}
```

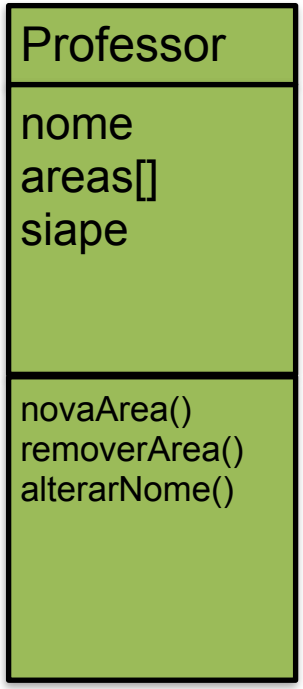


Como se cria um objeto Maria da classe Aluno?

```
public class ProgramaTeste1 {
    public static void main(String []args) {
        Aluno maria;

        maria = new Aluno();
        maria.nome = "Maria da Graça Souza";
        maria.curso = "Análise e Desenvolvimento de Sistemas (ADS)";
        maria.anoIngresso = 2013;
        maria.ehFormado = true;
        maria.matricula = 154090;
    }
}
```



Classe Professor

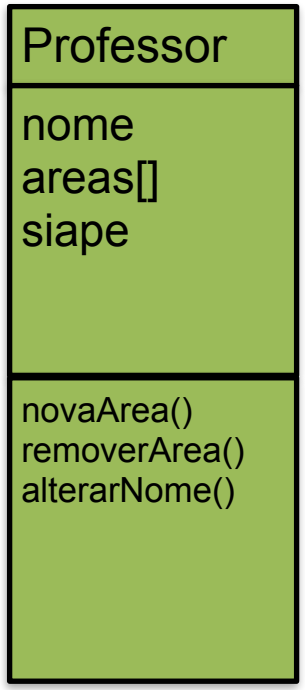



```
public class Professor {
    //Atributos
    String nome;
    String areas[];
    long siape;
    //Métodos
    void alterarNome (String novoNome) {
        this.nome = novoNome;
    }
    boolean novaArea(String area) {
        for (int i = 0; i < areas.length; i++)
            if (areas[i] == null) {
                areas[i] = area;
                return true;
            }
        return false;
    }
}
```

```
boolean removerArea(String area) {
    for (int i = 0; i < areas.length; i++)
        if (areas[i].equals(area)) {
            areas[i] = null;
            return true;
        }
    return false;
}
```

Como se cria um objeto Coelho da classe Professor?

```
public class ProgramaTeste2 {
    public static void main(String []args) {
        Professor coelho;

        coelho = new Professor();
        coelho.nome = “Rafael Vieira Coelho”;
        coelho.siape = 1804250;
        coelho.areas = new String[3];
        coelho.areas[0] = “Programação de Computadores";
        coelho.areas[1] = “Redes de Computadores”;
        coelho.areas[2] = “Segurança de Sistemas”;
    }
}
```



Como se cria um programa que armazena um vetor de alunos?

```
17  public class ProgramaTeste3 {
18
19      public static final BufferedReader br = new BufferedReader(new InputStreamReader(System.in))
20
21      public static void main(String[] args) throws IOException {
22          System.out.println("Escolha [1] para professor e [2] para aluno:");
23          int opcao = Integer.parseInt(br.readLine());
24
25          if (opcao == 1) {
26              Professor p = new Professor();
27              System.out.println("Nome:");
28              p.nome = br.readLine();
29              System.out.println("SIAPE:");
30              p.siape = Long.parseLong(br.readLine());
31              System.out.println("Quantas áreas?");
32              int quantAreas = Integer.parseInt(br.readLine());
33
34              p.areas = new String[quantAreas];
35              System.out.println("Informe as áreas:");
36              for (int i = 0; i < quantAreas; i++) {
37                  p.areas[i] = br.readLine();
38              }
39          } else {
```

Como se cria um programa que armazena um vetor de alunos?

```
39          } else {
40              Aluno a = new Aluno();
41              System.out.println("Nome:");
42              a.nome = br.readLine();
43              System.out.println("Curso:");
44              a.curso = br.readLine();
45              System.out.println("Matricula:");
46              a.matricula = Long.parseLong(br.readLine());
47              System.out.println("Ingresso:");
48              a.anoIngresso = Integer.parseInt(br.readLine());
49              a.ehFormado = false;
50          }
51      }
52  }
```

Tarefas

1) Crie um projeto chamado IFRS e adicione ao projeto as classes dadas como exemplo anteriormente (Aluno, Professor e as de teste)

2) Modularize as classes de teste.

3) Implemente as três classes abaixo.