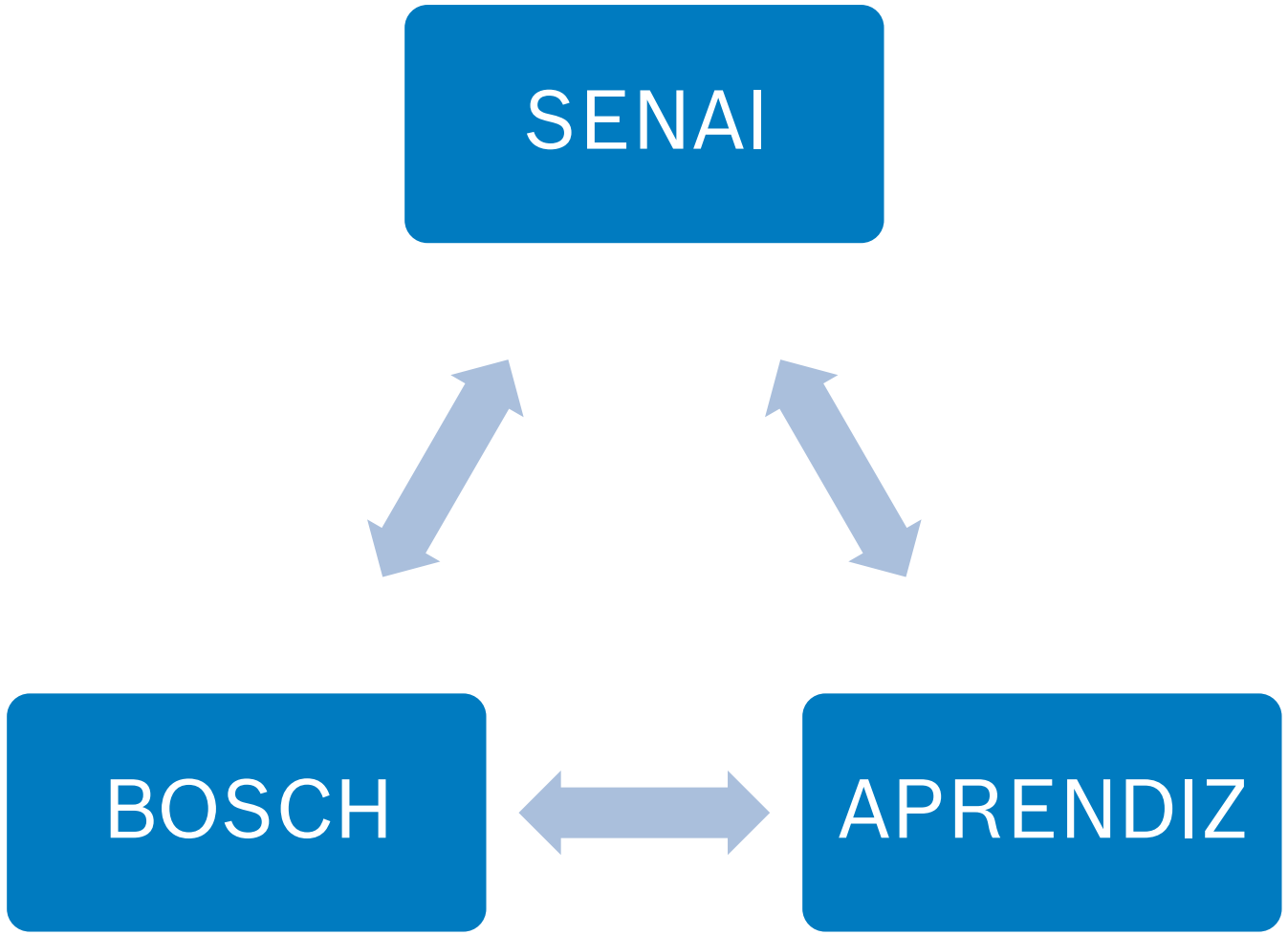
PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento



PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento

ENFERMEIRA

PACIENTE

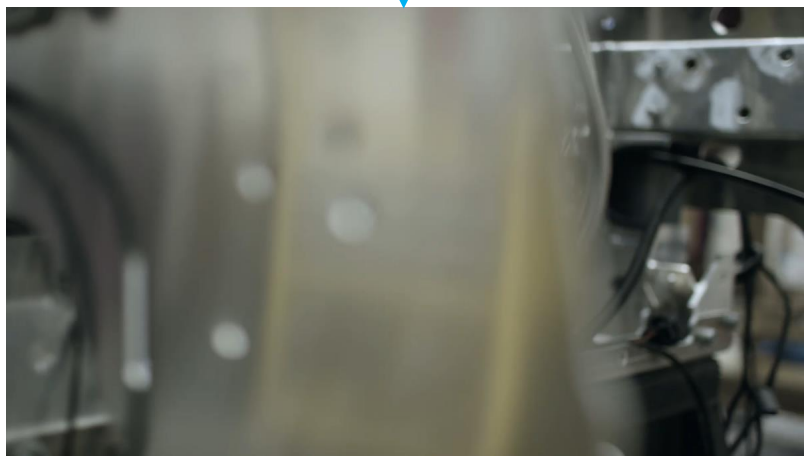
MÉDICO

PROGRAMAÇÃO ORIENTADA A OBJETOS

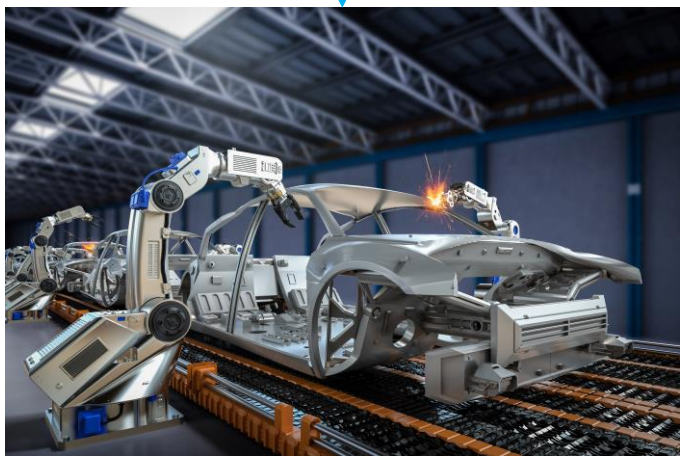
Composição

CARRO

PNEU



MOTOR



CHASSI

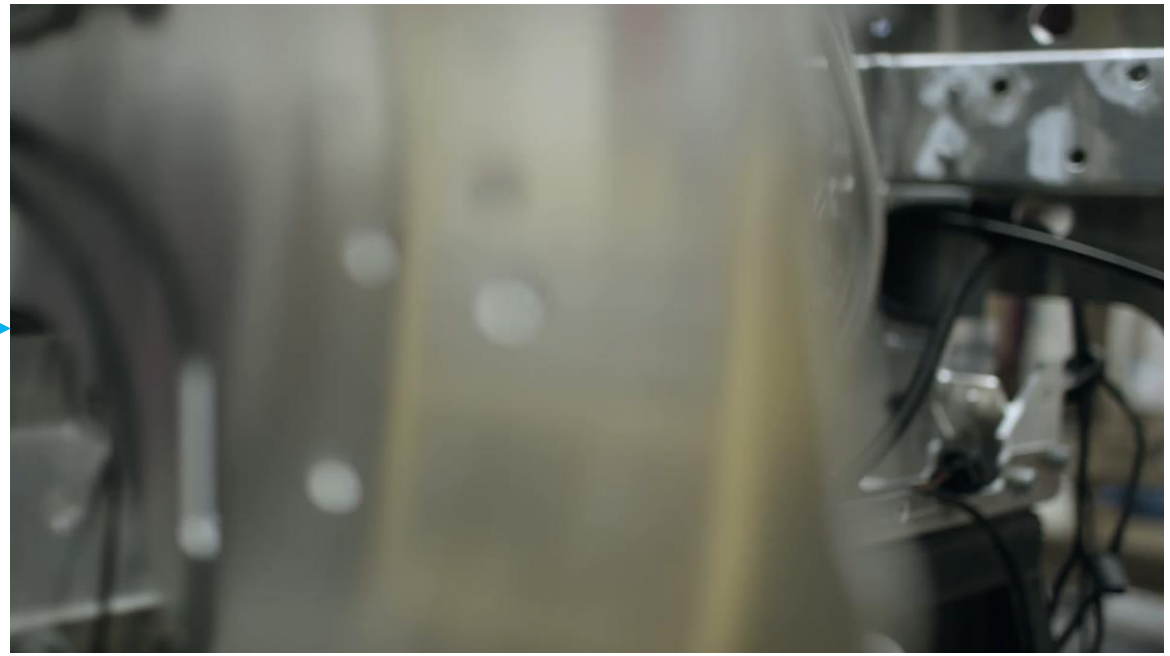
PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para um

1

CARRO

1



MOTOR

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para muitos

1

n

CARRO

MOTOR

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de muitos para muitos

TIO

SOBRINHO

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para um

```java
package bosch;

public class Motor {
    double fatorInjecao=1;
    boolean ligado=false;

    int giros()
    {
        if(!ligado) {
            return 0;
        }
        else{
            return (int) Math.round(fatorInjecao * 3000);
        }
    }

}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para um

Main.java | Motor.java | Carro.java

```java
package bosch;

public class Carro {
    Motor motor=new Motor();

    void acelerar(){
        motor.fatorInjecao+=0.4;
    }
    void frear()
    {
        motor.fatorInjecao-=0.4;
    }
    void ligar(){
        motor.ligado=true;
    }
    void desligar()
    {
        motor.ligado=false;
    }
    boolean estaLigado(){
        return motor.ligado;
    }

}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para um

```java
package bosch;

public class Main {

    public static void main(String[] args) {
        Carro carro = new Carro();
        System.out.println(carro.estaLigado());
        carro.ligar();
        System.out.println(carro.estaLigado());
        System.out.println(carro.motor.giros());
        carro.acelerar();
        carro.acelerar();
        System.out.println(carro.motor.giros());
        carro.frear();
        System.out.println(carro.motor.giros());
    }
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para muitos

```java
package bosch;

public class Item {
    String nome;
    int quantidade;
    double preco;
    Compra compra;

    Item(String nome, int quantidade,double preco){
        this.nome=nome;
        this.quantidade=quantidade;
        this.preco=preco;
    }
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para muitos

Main.java | Item.java | Compra.java

```java
package bosch;

import java.util.ArrayList;

public class Compra {
    String cliente;
    ArrayList<Item> itens=new ArrayList<Item>();

    void adicionarItem(Item item){
        this.itens.add(item);
        item.compra=this;
    }
     double obterValorTotal() {
        double total=0;
         for (Item item : itens) {
             total += item.quantidade * item.preco;
         }
         return total;
     }
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para muitos

Main.java | Item.java | Compra.java

```java
package bosch;

public class Main {

    public static void main(String[] args) {
    Compra compra1 = new Compra();
    compra1.cliente="João Pedro";
    compra1.adicionarItem(new Item( nome: "Notebook", quantidade: 1, preco: 5000));
    compra1.adicionarItem(new Item( nome: "Iphone", quantidade: 2, preco: 6000));
    compra1.adicionarItem(new Item( nome: "Amazon Echo", quantidade: 3, preco: 1200));
    System.out.println(compra1.itens.size());
    System.out.println(compra1.obterValorTotal());
    }
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de muitos para muitos

Main.java | Cursos.java | Aluno.java

```java
package bosch;

import java.util.ArrayList;

public class Aluno {
    String nome;
    ArrayList<Cursos> cursos=new ArrayList<Cursos>();
    Aluno(String nome){
        this.nome=nome;
    }
    void adicionarCurso(Cursos cursos)
    {
        this.cursos.add(cursos);
        cursos.alunos.add(this);
    }
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de muitos para muitos

Main.java | Cursos.java | Aluno.java

```java
package bosch;

import java.util.ArrayList;

public class Cursos {
    String nome;
    ArrayList<Aluno> alunos=new ArrayList<Aluno>();
    Cursos(String nome) { this.nome=nome; }

    void adicionarAluno(Aluno aluno){
        this.alunos.add(aluno);
        aluno.cursos.add(this);
    }

    ArrayList<String>obeterporCurso(){

        ArrayList<String>cadastrados=new ArrayList<~>();
        for (Aluno aluno:alunos) {
            cadastrados.add(aluno.nome);
        }
        return cadastrados;
    }
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de muitos para muitos

Main.java | Cursos.java | Aluno.java

```java
package bosch;

public class Main {

    public static void main(String[] args) {
        Aluno aluno1=new Aluno( nome: "João");
        Aluno aluno2=new Aluno( nome: "Maria");
        Aluno aluno3=new Aluno( nome: "Pedro");
        Cursos cursos1=new Cursos( nome: "Java");
        Cursos cursos2=new Cursos( nome: "Python");
        Cursos cursos3=new Cursos( nome: "VsDia");
        cursos1.adicionarAluno(aluno1);
        cursos1.adicionarAluno(aluno2);
        cursos2.adicionarAluno(aluno1);
        cursos2.adicionarAluno(aluno3);
        aluno1.adicionarCurso(cursos3);
        aluno2.adicionarCurso(cursos3);
        aluno3.adicionarCurso(cursos3);
        for (Aluno aluno: cursos3.alunos)
            {
                System.out.printf("Meu nome é %s e estour matriculado" +
                        " no curso %s\n",aluno.nome,cursos3.nome);
            }
        System.out.println(cursos3.alunos.get(0).nome);
        System.out.println(cursos3.obeterporCurso());
    }
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo de Delivery

- Um aplicativo de delivery é uma solução de gerenciamento de pedidos para restaurantes, visando ser uma opção acessível para os clientes pedirem comida em casa ou no trabalho. O aplicativo é configurado pelo gerente do restaurante e implementa as seguintes funcionalidades.

PROGRAMAÇÃO ORIENTADA A OBJETOS
Aplicativo de Delivery

- O aplicativo é configurado com os detalhes do restaurante: nome, CNPJ, e posição no mapa. Por simplicidade considere que a posição é um par de coordenadas (X, Y).
- O aplicativo permite o cadastro de usuários, que é feito informando nome, CPF, e endereço de entrega. Por simplicidade considere que o endereço é armazenado em forma de posição (X, Y).
- O aplicativo permite adicionar e remover itens ao cardápio do restaurante. Cada item contém nome, preço.
- O aplicativo permite imprimir o cardápio completo do restaurante.
- O aplicativo permite registrar pedidos dos usuários. Cada pedido pode incluir um ou mais itens do cardápio do restaurante
- O valor total do pedido é a soma dos preços atuais de cada item.

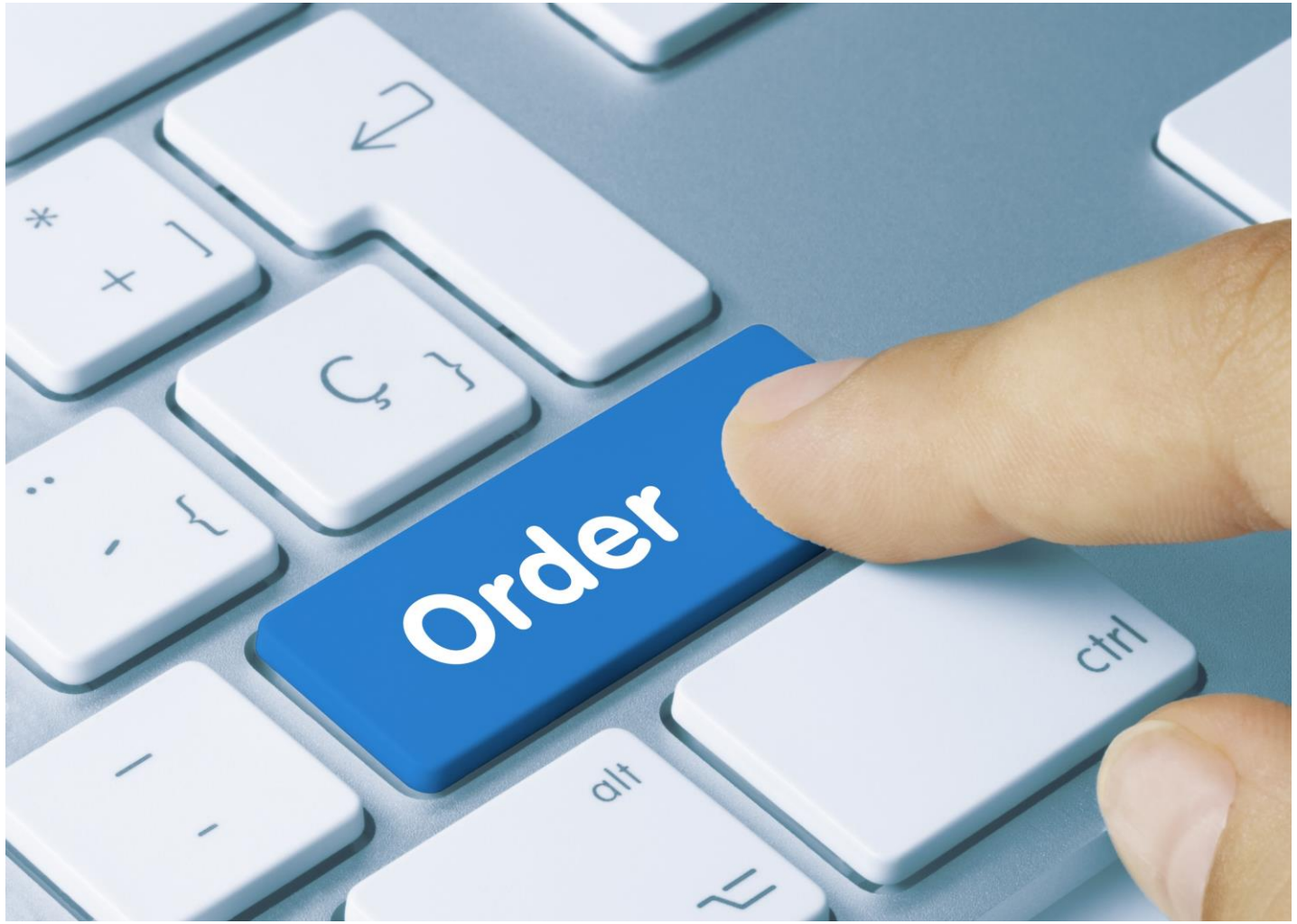
PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo de Delivery

Classe: Aplicativo **Instância:**	
Atributos	**Comportamentos**
Restaurantes	cadastrarRestaurante()
Usuários	cadastrarUsuario()
Pedidos	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo de Delivery

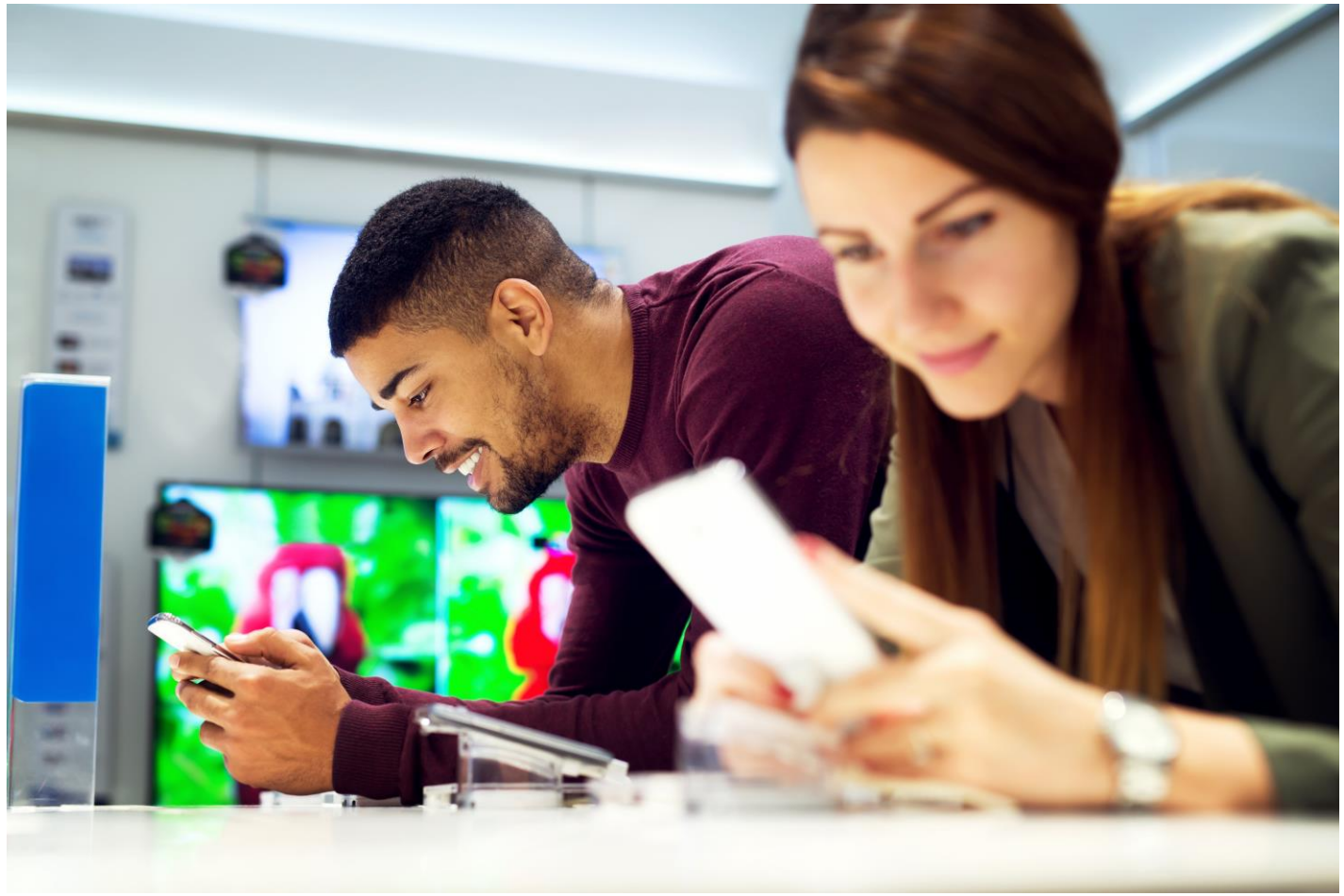


Classe: Pedido **Instância:**	
Atributos	**Comportamentos**
Restaurantes	fazerPedido()
Usuários	imprimirPedido()

PROGRAMAÇÃO ORIENTADA A OBJETOS
Aplicativo Delivery

Classe: Restaurante **Instância:**	
Atributos	**Comportamentos**
Nome	imprimirCardapio()
Preço	adicionarLanche()
	removerLanche()

PROGRAMAÇÃO ORIENTADA A OBJETOS
Aplicativo Delivery



Classe: Usuário **Instância: nome do Usuário**	
Atributos	**Comportamentos**
Nome	
Endereço	
CPF	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo Delivery

Classe: Lanche **Instância:**	
Atributos	**Comportamentos**
Nome	
Preço	