INTERFACE

Interface é um mecanismo que busca auxiliar na criação de classes, a utilização dela assegura que as classes que a implementam vão ter métodos semelhantes no quesito de funcionalidade, diferenciando que em cada classe eles serão implementados de formas diferentes. A interface garante como se fosse um contrato assinado que as classes que a utilizam deverão obrigatoriamente usar seus métodos.

1.1 Como declarar uma interface

Ao criar uma interface é importante lembrar que elas não possuem atributos, apenas métodos, ressaltando ainda, que estes só possuem assinatura e tipo, por exemplo int, double, boolean. Para declarar uma interface no Java é necessário estabelecer o modificador como public, dizer que é uma interface e estabelecer o nome da interface, já em C# é preciso apenas digitar interface + nome da interface, como mostra nas imagens a seguir.

Figura 1 - Interface no Java

Figura 2 - Interface em C#

Fonte: https://docs.microsoft.com/pt-br/dotnet/csharp/language-reference/keywords/interface

1.2 Como se implementa uma interface

E agora como utilizamos a interface criada? É muito simples, utilizando ainda os modelos das imagens anteriores, em Java basta declarar a classe normalmente e no final digitar “implements + nome da interface”, em C# é ainda mais fácil, é preciso escrever o nome da classe que vai receber a implementação seguido de “:” + nome da interface.

Para a utilização dos métodos em Java é necessário a palavra “@Override” seguida da implementação dos métodos.

Figura 3 - Implementação de Interface no Java

```java
public  class Celular implements AparelhoEletronico {
    @Override
    public boolean ligar(){
        return true;
    }

    @Override
    public boolean desligar(){
        return false;
    }

    @Override
    public void emitirSom(){
        System.out.println ("Hello moto");
    }
}
```

Para a utilização dos métodos em C# é necessário digitar “.” e logo em seguida o nome do método.

Figura 4 - Implementação de interface em C#

```csharp
class ImplementationClass : ISampleInterface
{
    // Explicit interface member implementation:
    void ISampleInterface.SampleMethod()
    {
        // Method implementation.
    }
```

Fonte: https://docs.microsoft.com/pt-br/dotnet/csharp/language-reference/keywords/interface

1.3 Diferença entre interface e classe abstrata

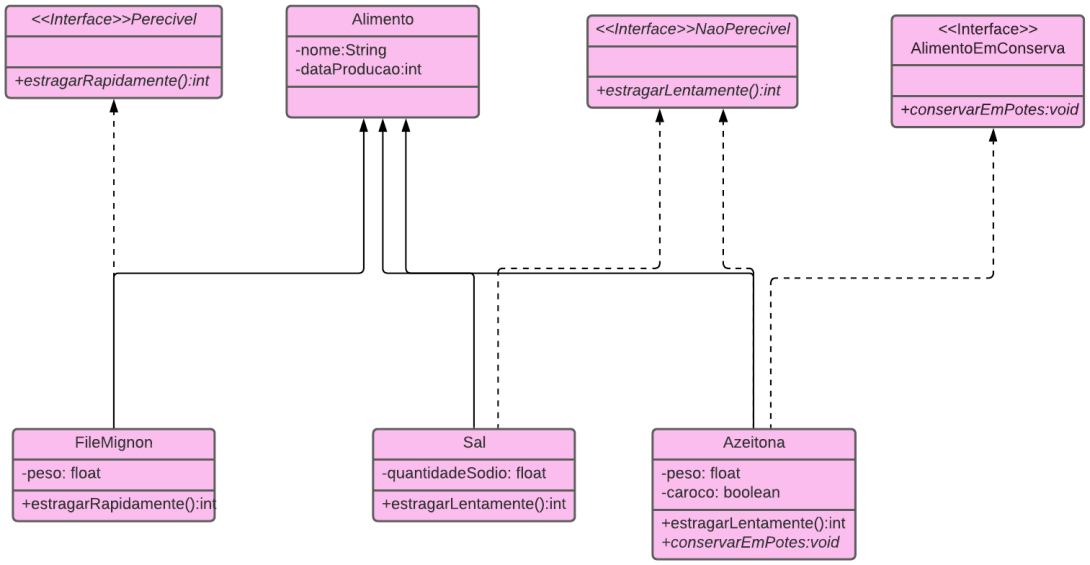
A diferença gritante entre interface e classe abstrata é o quanto uma classe pode implementá-las, uma classe só pode herdar uma classe abstrata, agora falando de interface não tem um número limite, outra diferença é a velocidade, interfaces são mais lentas que classe abstrata, pois precisam de tarefas para encontrar o método correspondente que a classe está implementando. Nas classes abstratas também é possível escrever lógicas, já na interface só é possível colocar a assinatura do método.

1.4 Lógica da interface

Para entender melhor a lógica da interface vamos usar um exemplo concreto:

Vamos supor que em um sistema é dividido os alimentos em perecíveis (carnes, leite…) e não perecíveis (milho, açúcar, sal..) , entretanto em uma determinada ocasião é necessário estabelecer também, os que são guardados em conserva e que não são perecíveis (azeitona, salsicha), poderíamos pensar que criando uma classe e depois fazendo um extends resolveria o problema, entretanto no Java não é possível estender duas classes (em algumas linguagens podem, como o C++), então para resolver esse problema podemos criar uma interface chamada AlimentoEmConserva.

A UML (Unified Modeling Language) então ficaria dessa forma:



Abaixo é possível ver a classe abstrata mãe “Alimento” implementada no Java

```java
package Arquivos.Interface.Comida;

public abstract class Alimento {

    private String nome;
    private int dataProducao;

    public int getFabricacao() {
        return dataProducao;
    }

    public void setFabricacao(int fabricacao) {
        this.dataProducao = dataProducao;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

}
```

E logo abaixo estão as interfaces, a Perecivel e a NaoPerecivel, respectivamente.

```java
package Arquivos.Interface.Comida;

public interface Perecivel {
     int estragarRapidamente();
}

```

```java
package Arquivos.Interface.Comida;

public interface NaoPerecivel  {
    int estragarLentamente();
}
```

O FileMignon e o Sal, respectivamente, implementam as interfaces Perecivel e NaoPerecivel, e ainda se estendem da classe mãe Alimento.

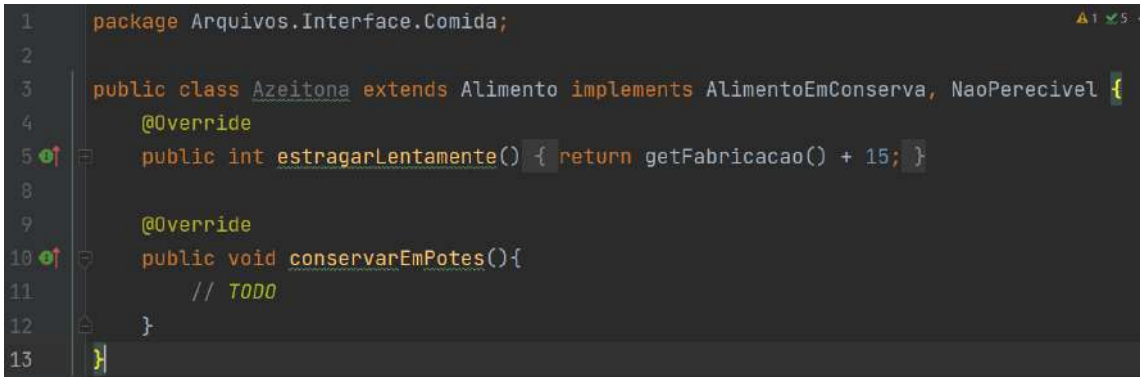
```java
package Arquivos.Interface.Comida;

public class FileMignon extends Alimento implements Perecivel {
    @Override
    public int estragarRapidamente() { return getFabricacao() + 5; }
}
```

```java
package Arquivos.Interface.Comida;

public class Sal extends Alimento implements NaoPerecivel{
    @Override
    public int estragarLentamente() { return getFabricacao() + 365; }
}
```

A seguir é mostrado a declaração da interface AlimentoEmConserva

```java
package Arquivos.Interface.Comida;

public interface AlimentoEmConserva {
    void conservarEmPotes();
}
```

E como foi mostrado na UML, a classe Azeitona implementa a interface AlimentoEmConserva e se estende da classe Alimento.



```java
package Arquivos.Interface.Comida;

public class Azeitona extends Alimento implements AlimentoEmConserva, NaoPerecivel {
    @Override
    public int estragarLentamente() { return getFabricacao() + 15; }

    @Override
    public void conservarEmPotes(){
        // TODO
    }
}
```

Projeto no Replit

https://replit.com/@gusttavosoares/lg2-artigo-interface

Conclusão

No Java e C# não é possível fazer o uso de herança múltiplas com classe, como no C++, para resolver esse problema é necessário utilizar a interface. Ela garante como um tratado que os métodos descritos nela vão ser usados nas classes que chamou a interface.

Referências

https://docs.microsoft.com/pt-br/dotnet/csharp/language-reference/keywords/interface
http://www.macoratti.net/net_ica1.htm
https://www.devmedia.com.br/entendendo-interfaces-em-java/25502
https://www.caelum.com.br/apostila/apostila-java-orientacao-objetos.pdf
https://www.devmedia.com.br/interfaces-x-classes-abstratas/13337
https://www.youtube.com/watch?v=6uLLfRNgRA4&t=536s