**extend**

* **Uso principal**: Se utiliza para establecer **herencia** entre clases o entre interfaces.
* **Herencia simple**: Una clase solo puede extender a **una única clase** (Java no soporta herencia múltiple de clases).
* **Relación entre clases**: Una clase que extiende a otra hereda todos sus métodos y atributos no privados.
* **Relación entre interfaces**: Una interfaz puede extender múltiples interfaces.

**implements**

* **Uso principal**: Se utiliza cuando una clase implementa una o más **interfaces**.
* **Múltiples interfaces**: Una clase puede implementar múltiples interfaces, permitiendo algo similar a la herencia múltiple.
* **Relación contrato**: Obliga a la clase que implementa la interfaz a proporcionar definiciones para todos los métodos declarados en la interfaz.

**Diferencias principales**

| **Aspecto** | **extend** | **implements** |
| --- | --- | --- |
| **Relación** | Clase-clase o interfaz-interfaz | Clase-interfaz |
| **Cantidad permitida** | Una clase puede extender solo a una clase, pero una interfaz puede extender múltiples interfaces. | Una clase puede implementar múltiples interfaces. |
| **Herencia vs Contrato** | Proporciona herencia de implementación. | Obliga a cumplir un contrato al implementar métodos. |
| **Métodos abstractos** | Puede heredar métodos abstractos o concretos. | Solo hereda métodos abstractos. |

**Similitudes**

1. Ambas palabras clave ayudan a reutilizar código y definir relaciones entre clases e interfaces.
2. Tanto extend como implements soportan polimorfismo.
3. Ambas afectan la estructura jerárquica de un programa en Java.