Was ist das?

`Cloneable` sie ist eine sogenannte **Marker-Schnittstelle** (Markierungsinterface). Das bedeutet, dass sie keine Methoden enthält, aber sie dient als Signal an das Java-Laufzeitsystem (JVM), dass eine Klasse das **Klonen** von Objekten über die Methode `clone()` unterstützen soll.

Die **`clone()`-Methode** selbst befindet sich tatsächlich in der Klasse `Object`. Aber warum sollen wir dann `Cloneable` implementieren, obwohl die Methode in `Object` definiert ist?

Warum `Cloneable` implementieren?

Der Zweck der **`Cloneable`-Schnittstelle** ist es, eine Art "Erlaubnis" zu erteilen, dass eine Klasse durch die `clone()`-Methode von `Object` klonbar ist. Wenn du versuchst, die `clone()`-Methode für eine Klasse aufzurufen, die **nicht** `Cloneable` implementiert, wirft die JVM eine **`CloneNotSupportedException`**.

Mit anderen Worten: Wenn eine Klasse die `Cloneable`-Schnittstelle **nicht** implementiert, signalisiert die JVM, dass es **nicht erlaubt ist**, diese Klasse zu klonen, auch wenn die Methode `clone()` von `Object` verfügbar ist.

Wie funktioniert das?

Die **`clone()`-Methode** in der Klasse `Object` funktioniert auf diese Weise:

- Sie ist eine **geschützte Methode** (`protected`), die du in einer abgeleiteten Klasse überschreiben musst, wenn du sie öffentlich oder anderweitig zugänglich machen willst.
- Sie prüft, ob das Objekt, das geklont werden soll, die Schnittstelle **`Cloneable`** implementiert.
    - Wenn das Objekt **`Cloneable`** implementiert, wird es geklont.
    - Wenn das Objekt **nicht `Cloneable`** implementiert, wird eine **`CloneNotSupportedException`** ausgelöst.

Warum ist `Cloneable` leer?

`Cloneable` ist eine **Marker-Schnittstelle**, was bedeutet, dass sie keine Methoden hat und nur als **Markierung** dient, um anzuzeigen, dass Objekte dieser Klasse geklont werden dürfen.

- Solche Schnittstellen sind nicht selten in der Java-Welt. Weitere Beispiele für Marker-Schnittstellen sind **`Serializable`** (für Serialisierung) und **`Remote`** (für Remote-Aufrufe).
- **Marker-Schnittstellen** ändern das Verhalten von Objekten zur Laufzeit, ohne dass Methoden hinzugefügt werden. Sie signalisieren der JVM oder dem Compiler, dass besondere Operationen (in diesem Fall Klonen) erlaubt sind.

Fazit:

1. **`Cloneable`** ist eine **Marker-Schnittstelle**, die anzeigt, dass die Klasse klonbar ist.
2. Die **`clone()`-Methode** ist in der Klasse **`Object`** definiert, aber sie kann nur verwendet werden, wenn die Klasse `Cloneable` implementiert.
3. Wenn du versuchst, `clone()` auf einer Klasse aufzurufen, die nicht `Cloneable` implementiert, wird eine **`CloneNotSupportedException`** geworfen.
4. Der Klonvorgang in Java kann **flach** (shallow copy) oder **tief** (deep copy) sein, je nachdem, wie du ihn implementierst.