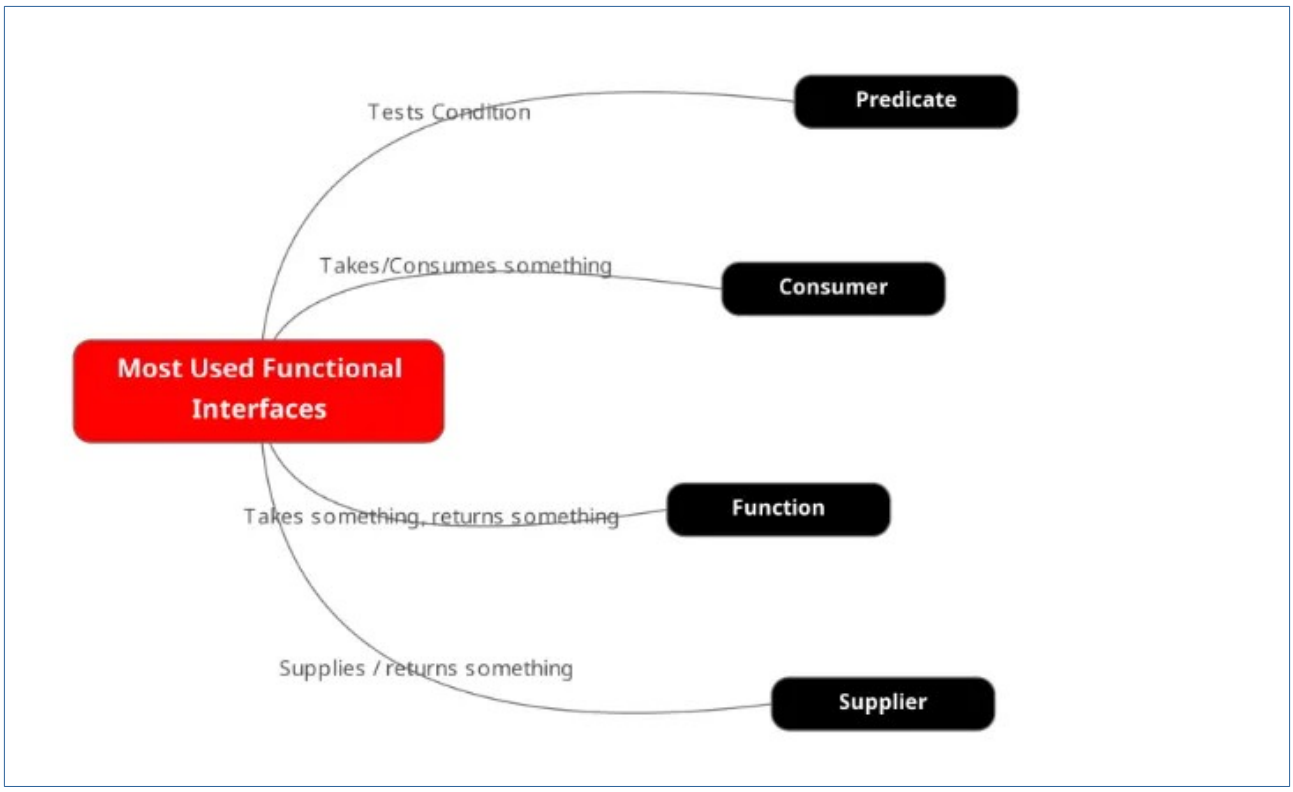

- **Predicate:** Testen einer Bedingung. => **boolean test(T t)**
- **Consumer:** Konsumiert einen Wert, führt eine Aktion aus, gibt aber nichts zurück. => **void accept(T t)**
- **Function:** Wandelt einen Wert in einen anderen um. => **R void accept(T t)**
- **Supplier:** Erzeugt oder liefert einen Wert ohne Eingabeparameter. => **T get()**

1. Predicate

- **Aufgabe:** Ein `Predicate` testet eine Bedingung und gibt einen `boolean`-Wert (`true` oder `false`) zurück.
- **Verwendung:** Wird häufig verwendet, um Filterbedingungen zu definieren, z. B. in `filter`-Methoden.
- **Methode:** `boolean test(T t)`
- **Beispiel:**

```
Predicate<Integer> isEven = x -> x % 2 == 0;
System.out.println(isEven.test(4)); // Gibt true zurück, da 4 gerade ist
```

- **Anwendung:** Zum Beispiel beim Filtern von Elementen in einem Stream, um nur Elemente zurückzugeben, die eine bestimmte Bedingung erfüllen.

2. Consumer

- **Aufgabe:** Ein `Consumer` akzeptiert ein Argument und führt eine Operation darauf aus, ohne etwas zurückzugeben.
- **Verwendung:** Wird häufig für Operationen verwendet, die eine Aktion ausführen, wie das Drucken oder Bearbeiten von Daten, ohne ein Ergebnis zurückzugeben.
- **Methode:** `void accept(T t)`
- **Beispiel:**

```
Consumer<String> print = s -> System.out.println(s);
print.accept("Hello, World!"); // Gibt "Hello, World!" auf der Konsole aus
```

- **Anwendung:** Zum Beispiel, um alle Elemente einer Liste auszugeben oder Operationen auf jedem Element eines Streams auszuführen.

3. Function

- **Aufgabe:** Eine `Function` nimmt ein Argument und gibt ein Ergebnis zurück. Sie nimmt also etwas und liefert etwas zurück.
- **Verwendung:** Wird oft verwendet, um Werte in andere Werte umzuwandeln, z. B. das Mapping eines Wertes auf einen anderen.
- **Methode:** `R apply(T t)`
- **Beispiel:**

```
Function<String, Integer> stringLength = s -> s.length();
System.out.println(stringLength.apply("ChatGPT")); // Gibt 7 zurück
```

- **Anwendung:** Zum Beispiel, um Elemente in einem Stream zu transformieren, indem man sie mit einer bestimmten Logik umwandelt.

4. Supplier

- **Aufgabe:** Ein `Supplier` liefert ein Ergebnis ohne Eingabeparameter zurück. Es liefert oder „versorgt“ einen Wert.
- **Verwendung:** Wird verwendet, um Werte zu generieren oder bereitzustellen, oft für nicht deterministische oder statische Werte.
- **Methode:** `T get()`
- **Beispiel:**

```
Supplier<Double> randomValue = () -> Math.random();
System.out.println(randomValue.get()); // Gibt eine zufällige Zahl zwischen 0 und 1 zu
```

- **Anwendung:** Zum Beispiel, um Standardwerte oder zufällige Werte zu erzeugen.