**BUILT-INT FUNCTIONAL INTERFACES**

Burada göreceğimiz interfaceler java.util.function paketi altında yer almaktadır.

Hepsi functional interfacedir.

**Predicate**

* Doğrulamak, karşılaştırma belirtimi gibi anlamlara gelmektedir.
* Return type olarak boolean değer dönen bir abstract metota sahiptir.
* Metotun grameri **boolean test(T t)** şeklindedir.
* Stream içerisinde **filter()** metotunda kullanılır.

Predicate arayüzü aşağıdaki gibi tanımlanabilir;

Yöntem 1:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Birden fazla predicate’i and ve or metodu ile birleştirebilriz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

List Collection’ı içerisinde ki removeIf metodu da predicate bir parametre alır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Predicate Primitive Version**

IntPredicate, LongPredicate, DoublePredicate versiyonları vardır.

Primitive tipler performans acısından daha hızlıdırlar.

Wrapper sınıf olduğu için boxing unboxing işlemleri performans acısından olumsuz etkiler.

**Örnek**:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**BiPredicate Version**

Yapısı şu şekildedir.

BiPredicate<T, U> boolean test(T t, U u);

Örnekle açıklanmıştır.

**Consumer**

* Tüketici, tüketmek gibi anlamlara gelmektedir.
* Return type olarak void değer dönen bir abstract metota sahiptir.
* Metotun grameri **void accept(T t);** şeklindedir.
* Stream içerisinde **foreach**() metotunda kullanılır.

Consumer arayüzü aşağıdaki gibi tanımlanabilir:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Consumer Primitive Version**

IntConsumer, LongConsumer, DoubleConsumer versiyonları vardır.

Primitive tipler performans acısından daha hızlıdırlar.

Wrapper sınıf olduğu için boxing unboxing işlemleri performans acısından olumsuz etkiler.

Örnek:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Consumer Binary Version**

Yapısı şu şekildedir.

BiConsumer<T, U> void accept(T t, U u)

Örnekle açıklanmıştır.

**Function**

* Stream içerinde yer alan **map()** metotunda kullanılır.
* Function<T, R> şeklinde iki parametre alır.
* R apply(T t); metoduyla ilk aldığı parametreyi değişken tip olarak ikinci parametreyi return type olarak kullanır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Function Primitive Version**

IntFunction, DoubleFunction, LongFunction olmak üzere primitive tipleri vardır.

Primitive tipler performans acısından daha hızlıdırlar.

Wrapper sınıf olduğu için boxing unboxing işlemleri performans acısından olumsuz etkiler.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Function Binary Version**

Yapısı şu şekildedir.

BiFunction<T, U, R> R apply(T t, U u);

Örnekle açıklanmıştır.

**Supplier**

* Tedarikçi, sağlayan gibi anlamlara karşılık gelmektedir.
* Generic type tipinde bir çıktı üretir.
* Stream üzerinde **generate**() metodunda kullanılır.
* Generate metodundan sonra **limit**() vermezsek sonsuz döngüye girebilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

nextInt method static olmadığından instance üzerinden kullanıldı.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Supplier Primitive Version**

IntSupplier, DoubleSupplier, LongSupplier, BooleanSupplier olmak üzere primitive tipleri vardır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Parametre almadığı için binary versiyonu yoktur.

**Unary operator**

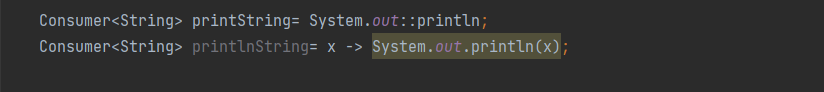
Function interfacesini kalıtmakta.

replaceAll() method parametre olarak Unary operatoru almakta.

Primitive versiyonları mevcuttur.

**Method Referans**

Aşağıdaki iki kullanımda aynıdır.



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Not:** Static olmayan metotlarda instance üzerinden metot referans uygulanır. Bir istisna hariç.

Açıklama:

Predicate<String> boolean test(T t)

T-> String

String sınıfı içerisinde parametresiz Boolean dönen metotları static olmasa bile metot referans şeklinde kullanabiliriz.