**Core Functional Interfaces**

| **Interface** | **Method Signature** | **Description** |
| --- | --- | --- |
| **Function<T,R>** | R apply(T t) | Takes T, returns R |
| **BiFunction<T,U,R>** | R apply(T t, U u) | Takes T & U, returns R |
| **Consumer<T>** | void accept(T t) | Takes T, returns nothing |
| **BiConsumer<T,U>** | void accept(T t, U u) | Takes T & U, returns nothing |
| **Supplier<T>** | T get() | Takes nothing, returns T |
| **Predicate<T>** | boolean test(T t) | Takes T, returns boolean |
| **BiPredicate<T,U>** | boolean test(T t, U u) | Takes T & U, returns boolean |
| **UnaryOperator<T>** | T apply(T t) | Takes T, returns T |
| **BinaryOperator<T>** | T apply(T t, T u) | Takes T & T, returns T |

**Primitive Specializations**

* **IntFunction<R>**, **LongFunction<R>**, **DoubleFunction<R>**
* **IntConsumer**, **LongConsumer**, **DoubleConsumer**
* **IntSupplier**, **LongSupplier**, **DoubleSupplier**
* **IntPredicate**, **LongPredicate**, **DoublePredicate**
* **IntUnaryOperator**, **LongUnaryOperator**, **DoubleUnaryOperator**
* **IntBinaryOperator**, **LongBinaryOperator**, **DoubleBinaryOperator**
* **ToIntFunction<T>**, **ToLongFunction<T>**, **ToDoubleFunction<T>**
* **ToIntBiFunction<T,U>**, **ToLongBiFunction<T,U>**, **ToDoubleBiFunction<T,U>**

**Example Usage**

Function<String, Integer> strLength = s -> s.length();

Predicate<Integer> isEven = n -> n % 2 == 0;

Consumer<String> printer = s -> System.out.println(s);

Supplier<Double> random = () -> Math.random();

**Note:**  
A functional interface is any interface with a single abstract method (SAM), and you can create your own as well using @FunctionalInterface.