Java Functional Interfaces Cheat Sheet

Predicate<T>

- Method: boolean test(T t)
- Description: Takes one input, returns true/false. Used for filtering.
- Example: Predicate<String> isShort = s -> s.length() < 5;

System.out.println(isShort.test("Hi")); // true

Function<T, R>

- Method: R apply(T t)
- Description: Takes one input, returns a result. Used for mapping/transformation.
- Example: Function<Integer, String> toStr = i -> "Value: " + i;

System.out.println(toStr.apply(10)); // "Value: 10"

Consumer<T>

- Method: void accept(T t)
- Description: Takes one input, returns nothing. Used for side-effects (like printing).
- Example: Consumer<String> printer = s -> System.out.println("Hi " + s);

printer.accept("Alice"); // prints Hi Alice

Supplier<T>

- Method: T get()
- Description: Takes no input, returns a result. Used to lazily supply data.
- Example: Supplier<Double> random = () -> Math.random();

System.out.println(random.get());

UnaryOperator<T>

- Method: T apply(T t)
- Description: Same as Function, but input and output types are the same.
- Example: UnaryOperator<Integer> square = x -> x * x;

System.out.println(square.apply(4)); // 16

BinaryOperator<T>

- Method: T apply(T t1, T t2)
- Description: Takes 2 inputs of the same type, returns one. Used in reduce().

Java Functional Interfaces Cheat Sheet

- Example: BinaryOperator<Integer> sum = (a, b) -> a + b;

System.out.println(sum.apply(4, 5)); // 9

BiFunction<T, U, R>

- Method: R apply(T t, U u)
- Description: Takes 2 inputs, returns 1 result (e.g., combining values).
- Example: BiFunction<String, Integer, String> combiner = (s, i) -> s + i;

System.out.println(combiner.apply("Age: ", 30));

BiConsumer<T, U>

- Method: void accept(T t, U u)
- Description: Takes 2 inputs, returns nothing.
- Example: BiConsumer<String, Integer> printer = (s, i) -> System.out.println(s + i);

printer.accept("Score: ", 95);

BiPredicate<T, U>

- Method: boolean test(T t, U u)
- Description: Takes 2 inputs, returns true/false.
- Example: BiPredicate<String, Integer> checker = (s, i) -> s.length() == i;

System.out.println(checker.test("Java", 4)); // true