# Функциональные интерфейсы

* java.util.function
* @FunctionalInterface

DoublePredicate

* boolean test(double value)
* default DoublePredicate and(DoublePredicate other)
* default DoublePredicate negate()
* default DoublePredicate or(DoublePredicate other)

BiPredicate<T, U>

* boolean test(T t, U u)
* default BiPredicate<T, U> and(  
  BiPredicate<? super T, ? super U> other)
* default BiPredicate<T, U> negate()
* default BiPredicate<T, U> or(  
  BiPredicate<? super T, ? super U> other)

## Consumer<T>

* void accept(T t)
* default Consumer<T> andThen(Consumer<? super T> after)

IntConsumer

* void accept(int value)
* andThen()

DoubleConsumer

* void accept(double value)
* default DoubleConsumer andThen(DoubleConsumer after)

BiConsumer<T, U>

* void accept(T t, U u)
* default BiConsumer<T, U> andThen(  
  BiConsumer<? Super T, ? super U> after)

BooleanSupplier

* boolean getAsBoolean()

DoubleSupplier

* double getAsDouble()

## Function<T, R>

* R apply(T t)
* default <V> Function<V, R> compose(  
  Function<? super V, ? extends T> before)
* default <V> Function<T, V> andThen(  
  Function<? super R, ? extends V> after)
* static <T> Function<T, T> identity()

DoubleFunction<R>

* R apply(double value)

DoubleToIntFunction

* int applyAsInt(double value)

DoubleToLongFunction

* long applyAsLong(double value)

DoubleUnaryOperator

* double applyAsDouble(double operand)
* default DoubleUnaryOperator compose(  
  DoubleUnaryOperator before)
* default DoubleUnaryOperator andThen(  
  DoubleUnaryOperator after)
* static DoubleUnaryOperator identity()

BiFunction<T, U, R>

* R apply(T t, U u)
* default <V> BiFunction<T, U, V> andThen(  
  Function<? super R, ? extends V> after)

BinaryOperator<T>

* extends BiFunction<T,T,T>
* public static <T> BinaryOperator <T>minBy(  
  Comparator<? super T> comparator)
* public static <T> BinaryOperator<T> maxBy(  
  Comparator<? super T> comparator)

IntBinaryOperator

* int applyAsInt(int left, int right)

DoubleBinaryOperator

* double applyAsDouble(double left, double right)