Functional Programming

Functional Interfaces for Primitives

COMMON FUNCTIONAL INTERFACES FOR PRIMITIVES (1/2)

Functional Interface	Return Type	Abstract Method Signature
`BooleanSupplier`	`boolean`	`getAsBoolean()`
`DoubleSupplier`	`double`	`getAsDouble()`
`IntSupplier`	`int`	`getAsInt()`
`LongSupplier`	`long`	`getAsLong()`
`DoubleConsumer`	`void`	`accept(double a)`
`IntConsumer`	`void`	`accept(int a)`
`LongConsumer`	`void`	`accept(long value)`
`DoublePredicate`	`boolean`	`test(double value)`
`IntPredicate`	`boolean`	`test(int value)`
`LongPredicate`	`boolean`	`test(long value)`

COMMON FUNCTIONAL INTERFACES FOR PRIMITIVES (2/2)

Functional Interface	Return Type	Abstract Method Signature
`DoubleFunction<R>`	R	`apply(double value)`
`IntFunction<R>`	R	`apply(int value)`
`LongFunction<R>`	R	`apply(long value)`
`DoubleUnaryOperator`	double	`applyAsDouble(double value)`
`IntUnaryOperator`	int	`applyAsInt(int value)`
`LongUnaryOperator`	long	`applyAsLong(long value)`
`DoubleBinaryOperator`	double	`applyAsDouble(double v, double w)`
`IntBinaryOperator`	int	`applyAsInt(int v, int w)`
`LongBinaryOperator`	long	`applyAsLong(long v, long w)`

PRIMITIVE-SPECIFIC FUNCTIONAL INTERFACES (1/2)

Functional Interface	Return Type	Abstract Method Signature
`ToDoubleFunction<T>`	`double`	`applyAsDouble(T t)`
`ToIntFunction<T>`	`int`	`applyAsInt(T t)`
`ToLongFunction<T>`	`long`	`applyAsLong(T t)`
`ToDoubleBiFunction<T, U>`	`double`	`applyAsDouble(T t, U u)`
`ToIntBiFunction<T, U>`	`int`	`applyAsInt(T t, U u)`
`ToLongBiFunction<T, U>`	`long`	`applyAsLong(T t, U u)`

PRIMITIVE-SPECIFIC FUNCTIONAL INTERFACES (2/2)

Functional Interface	Return Type	Abstract Method Signature
`DoubleToIntFunction`	`int`	`applyAsInt(double value)`
`DoubleToLongFunction`	`long`	`applyAsLong(double value)`
`IntToDoubleFunction`	`double`	`applyAsDouble(int value)`
`IntToLongFunction`	`long`	`applyAsLong(int value)`
`LongToDoubleFunction`	`double`	`applyAsDouble(long value)`
`LongToIntFunction`	`int`	`applyAsInt(long value)`
`ObjDoubleConsumer<T>`	`void`	`accept(T t, double value)`
`ObjIntConsumer<T>`	`void`	`accept(T t, int value)`
`ObjLongConsumer<T>`	`void`	`accept(T t, long value)`