Working with Functional Interfaces

Jesper de Jong
Software Architect

@jesperdj www.jesperdj.com

Overview

- What exactly is a functional interface?
- The @FunctionalInterface annotation
- Common standard functional interfaces
- Functional composition
- Specialized standard functional interfaces

Understanding Functional Interfaces

Understanding Functional Interfaces

A functional interface is an interface with a single abstract method

```
interface Comparator<T> {
    int compare(T o1, T o2);
}
```

```
interface Runnable {
    void run();
}
```

```
interface FileFilter {
    boolean accept(File f);
}
```

```
interface ActionListener {
    void actionPerformed(ActionEvent e);
}
```

The @FunctionalInterface Annotation

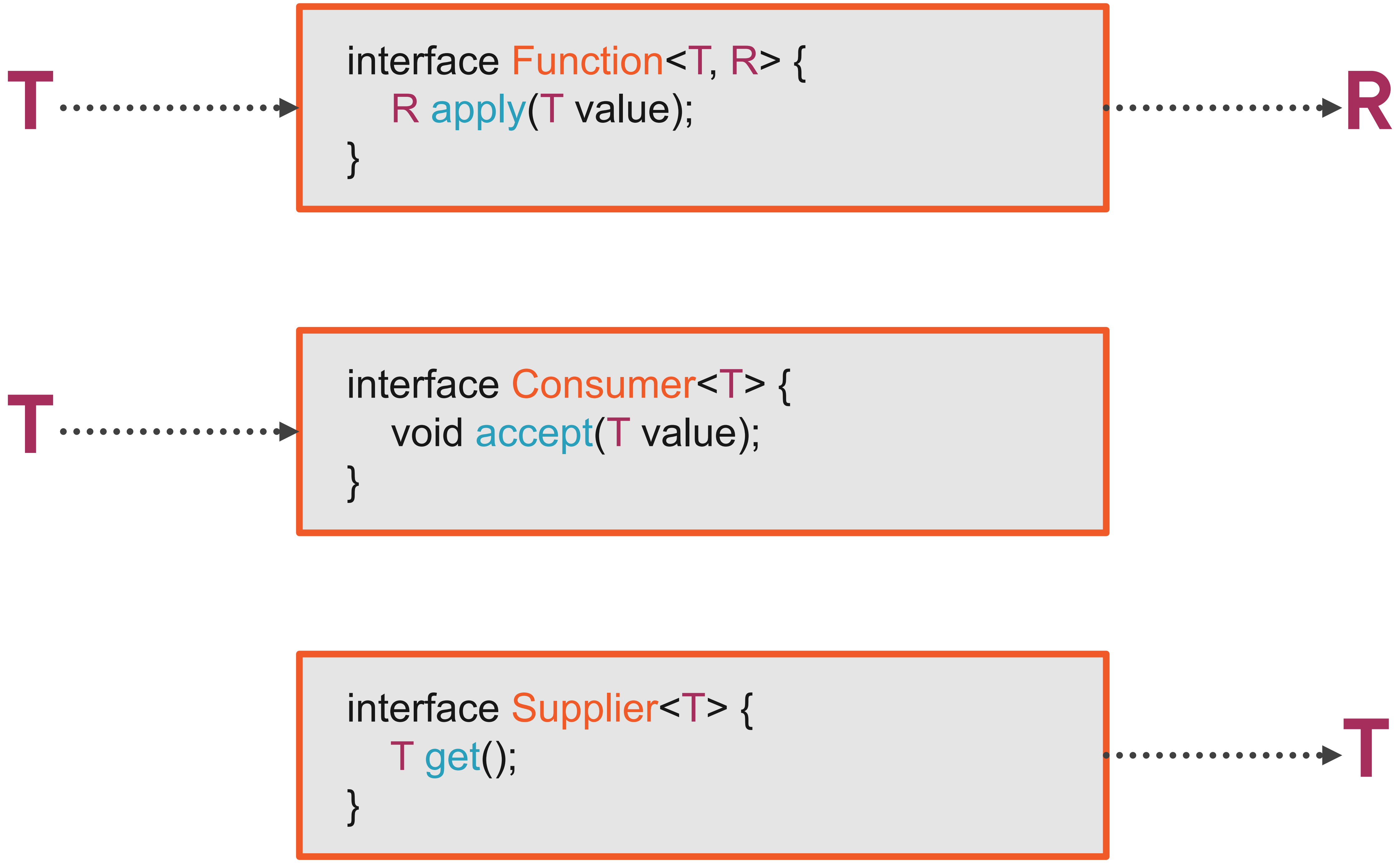
```java
@FunctionalInterface
interface ProductFilter {
    boolean test(Product product);
    void print(Product product);
}
```

Expresses that an interface is intented to be used as a functional interface

Not required

Common Standard Functional Interfaces

Common Standard Functional Interfaces

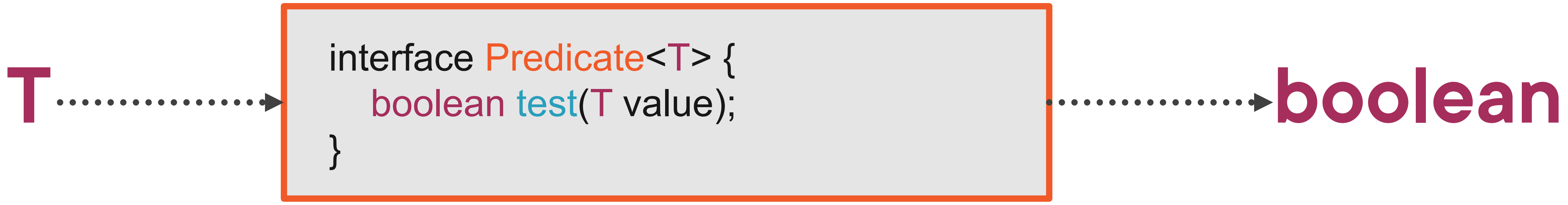


```
interface Function<T, R> {
    R apply(T value);
}
```

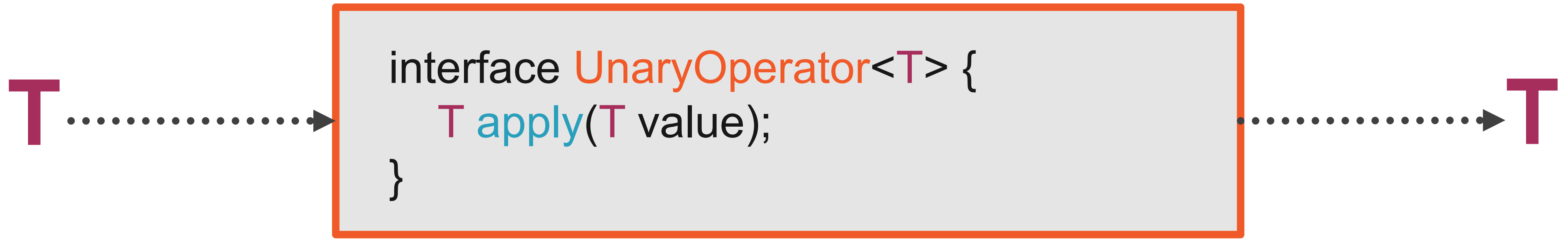
```
interface Consumer<T> {
    void accept(T value);
}
```

```
interface Supplier<T> {
    T get();
}
```

Common Standard Functional Interfaces

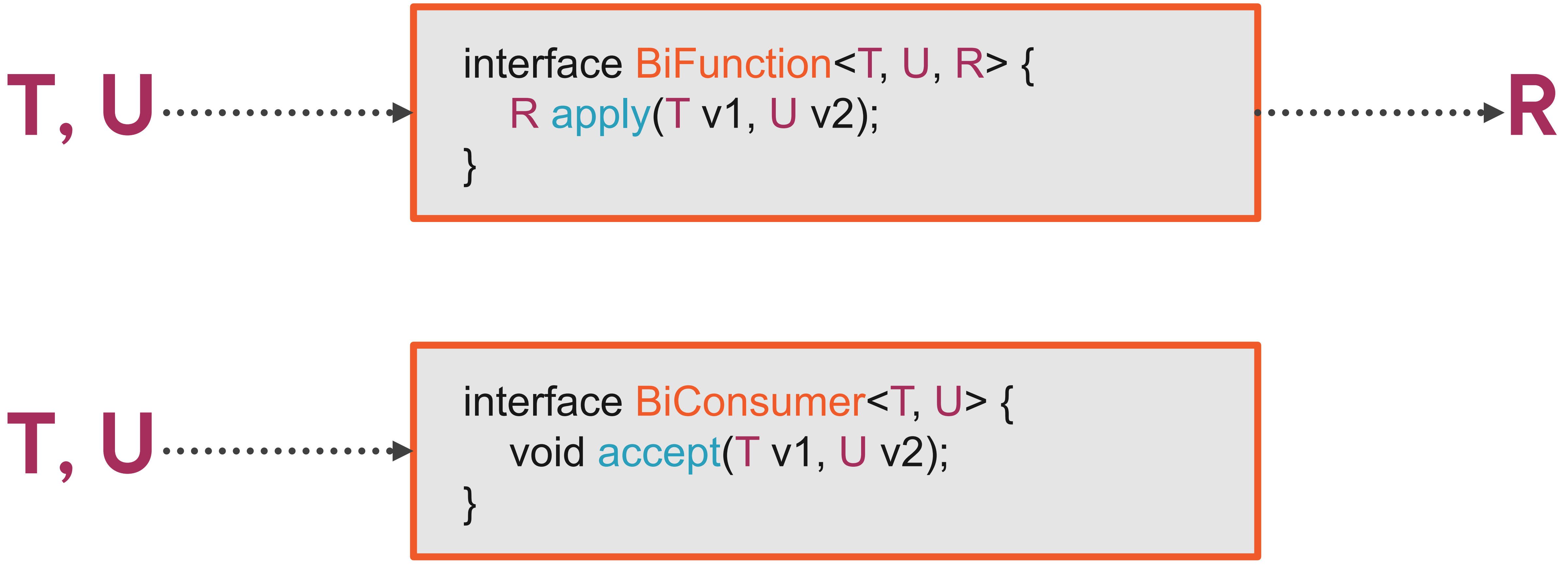


```
interface Predicate<T> {
    boolean test(T value);
}
```



```
interface UnaryOperator<T> {
    T apply(T value);
}
```

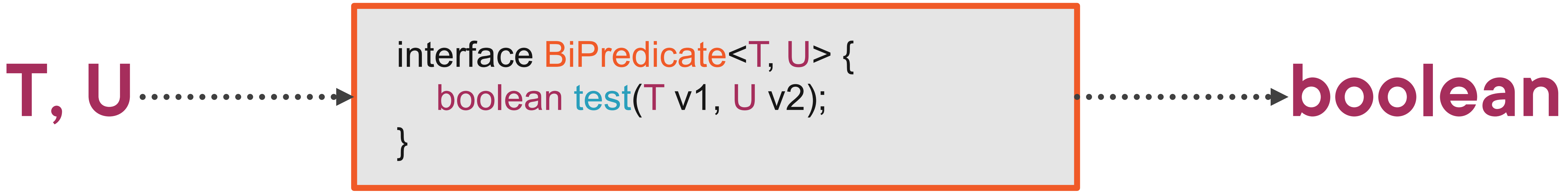
Common Standard Functional Interfaces



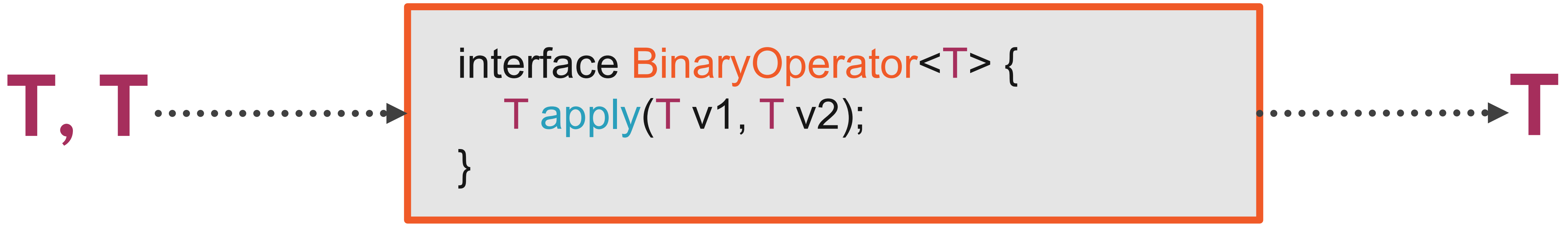
```
interface BiFunction<T, U, R> {
    R apply(T v1, U v2);
}
```

```
interface BiConsumer<T, U> {
    void accept(T v1, U v2);
}
```

Common Standard Functional Interfaces

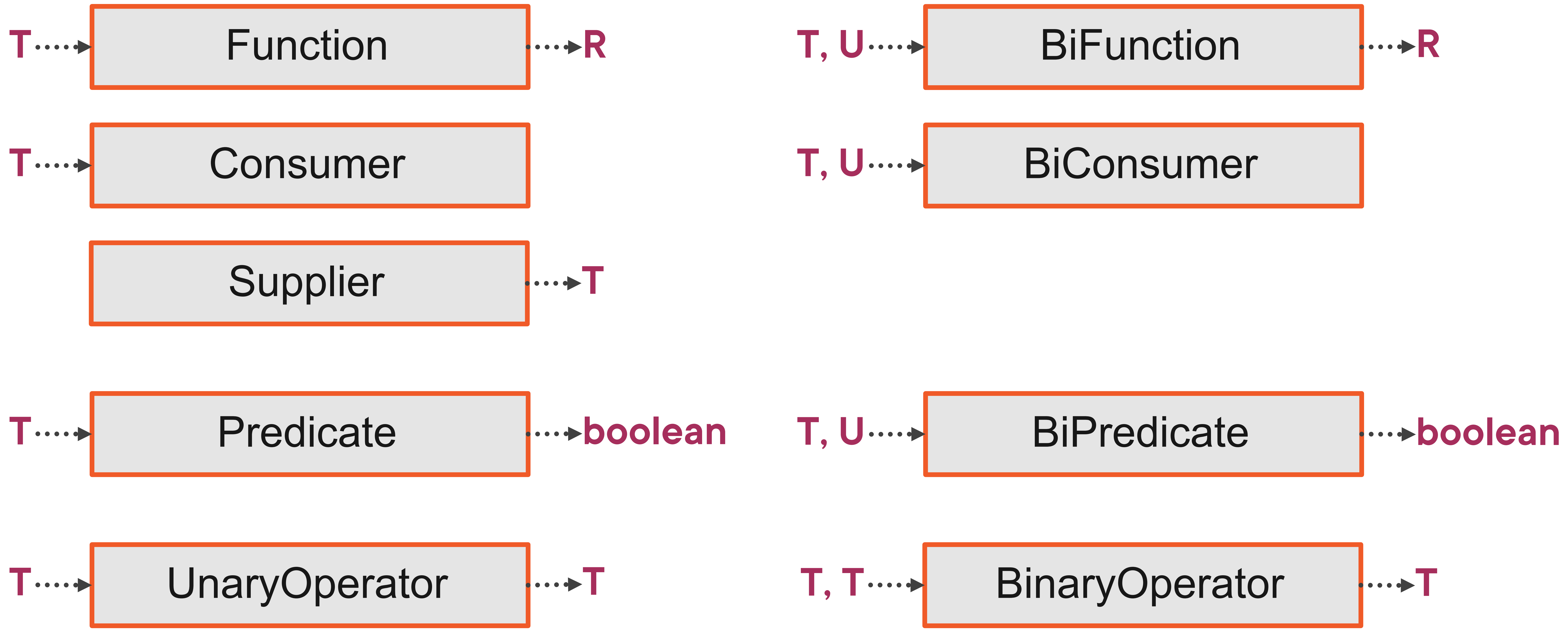


```
interface BiPredicate<T, U> {
    boolean test(T v1, U v2);
}
```



```
interface BinaryOperator<T> {
    T apply(T v1, T v2);
}
```

Common Standard Functional Interfaces

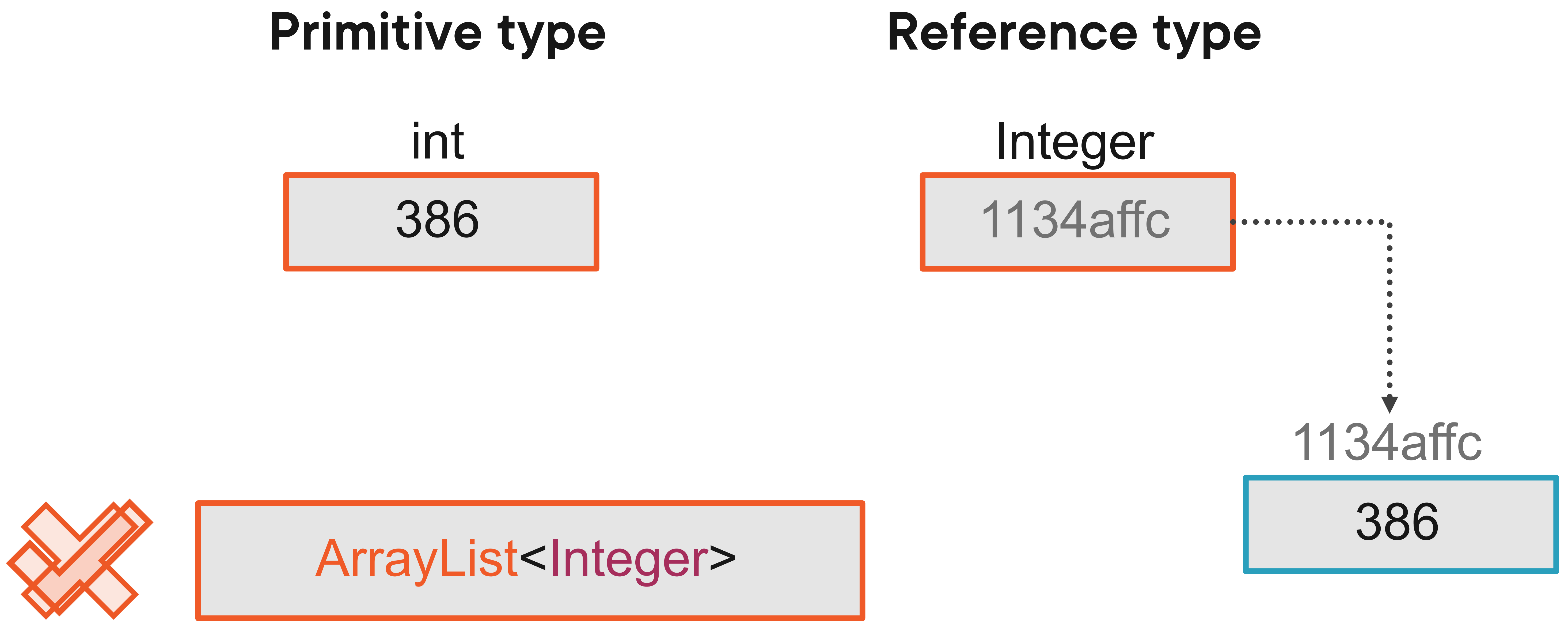


Practical Examples of Standard Functional Interfaces

Functional Composition

Specialized Standard Functional Interfaces

Primitive Types and Reference Types

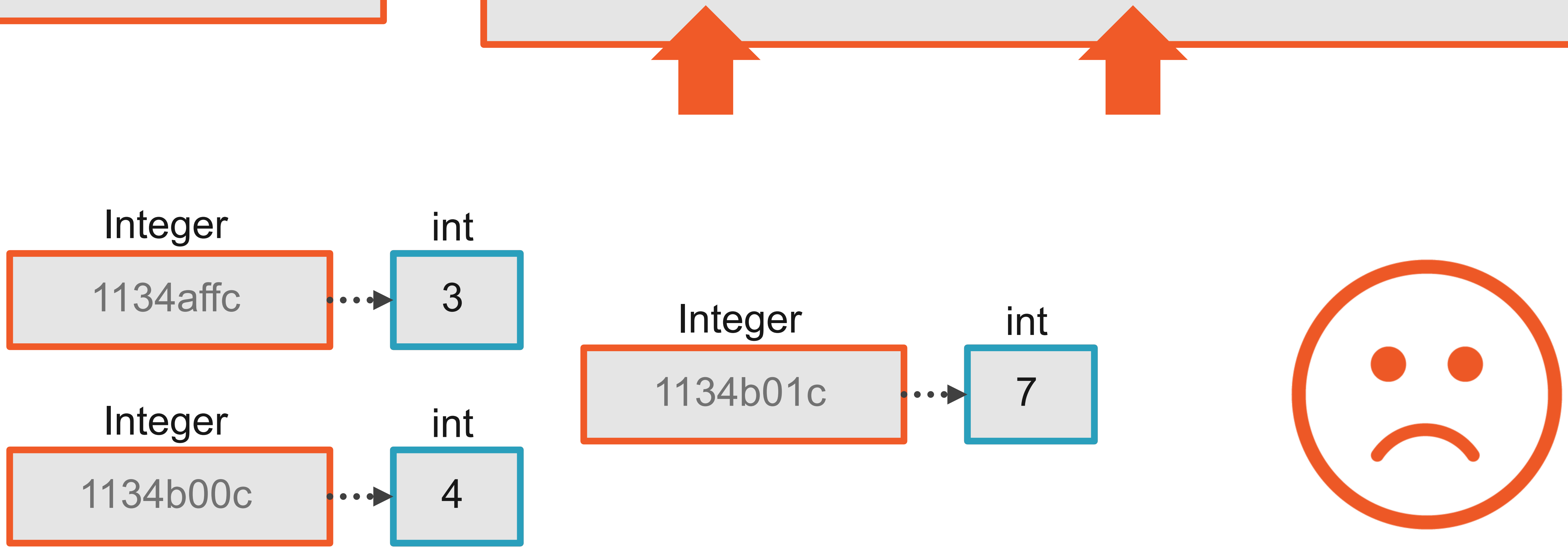


Specialized Standard Functional Interfaces

```
interface BinaryOperator<T> {
    T apply(T v1, T v2);
}
```

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

int result = sum.apply(3, 4);
```



Specialized Standard Functional Interfaces

```
interface IntBinaryOperator<T> {
    int applyAsInt(int v1, int v2);
}
```

```
interface BinaryOperator<T> {
    T apply(T v1, T v2);
}
```

Specialized Standard Functional Interfaces

byte
short
int
long
float
double
char
boolean

PrefixToSuffixInterface

IntFunction<R>	int ⋯⋯▸ R
LongToDoubleFunction	long ⋯⋯▸ double
ObjIntConsumer<T>	T, int ⋯⋯▸ void

Specialized Standard Functional Interfaces

XFunction<R>	XPredicate
XToYFunction	XConsumer
ToXFunction<T>	ObjXConsumer<T>
ToXBiFunction<T,U>	XSupplier
XUnaryOperator	XBinaryOperator

X, Y **= Int, Long, Double**

Extra: BooleanSupplier

Summary of Functional Interfaces

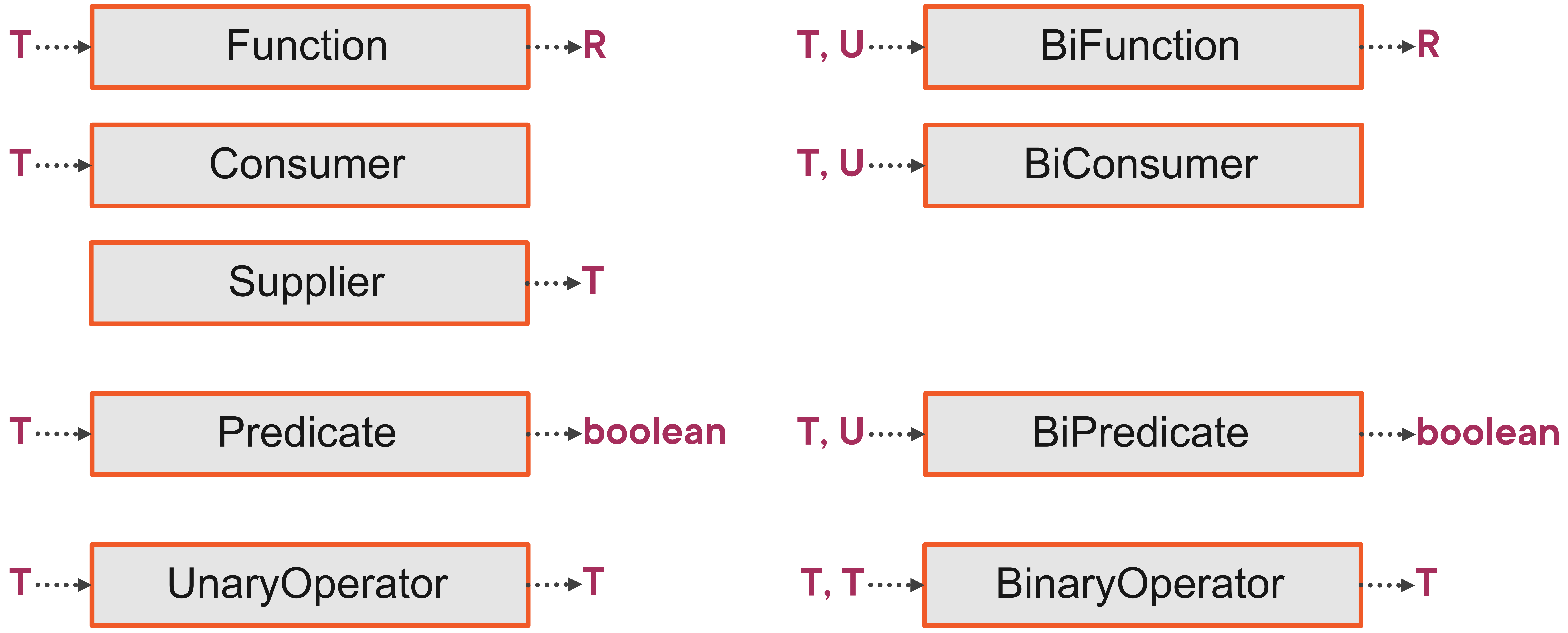
Functional Interfaces

```
@FunctionalInterface
interface Comparator<T> {
    int compare(T o1, T o2);

    default Comparator<T> reversed() { ... }
    static <...> naturalOrder() { ... }

    boolean equals(Object obj);
}
```

Common Standard Functional Interfaces



Functional Composition

Function	andThen	Function
Function	compose	Function
Predicate	and	Predicate

Specialized Standard Functional Interfaces

XFunction<R>	XPredicate
XToYFunction	XConsumer
ToXFunction<T>	ObjXConsumer<T>
ToXBiFunction<T,U>	XSupplier
XUnaryOperator	XBinaryOperator

X, Y **= Int, Long, Double**

Extra: BooleanSupplier

Up Next: Working with Streams – The Basics