**Functional Interfaces Cheat Sheet**

1. Consumer: Represents an operation that accepts a single input argument and returns no result.

public interface Consumer<T> {

void accept(T t);

}

1. BiConsumer: Represents an operation that accepts two input arguments and returns no result.

public interface BiConsumer<T,U>{

void accept(T t, U u)

}

1. Supplier: Represents a supplier of results.

public interface Supplier<T> {

T get();

}

1. Function: Represents a function that accepts one argument and produces a result.

public interface Function<T, R> {

R apply(T t);

}

1. BiFunction: Represents a function that accepts two arguments and produces a result.

public interface BiFunction<T,U,R>{

[R](http://docs.oracle.com/javase/8/docs/api/java/util/function/BiFunction.html) apply([T](http://docs.oracle.com/javase/8/docs/api/java/util/function/BiFunction.html" \o "type parameter in BiFunction) t, [U](http://docs.oracle.com/javase/8/docs/api/java/util/function/BiFunction.html) u)

}

1. Predicate: Represents a predicate (boolean-valued function) of one argument.

public interface Predicate<T> {

boolean test(T t);

}

1. BiPredicate: Represents a predicate (boolean-valued function) of two arguments.

public interface BiPredicate<T, U> {

boolean test(T t, U u);

}

1. Comparator: A comparison function, which imposes an ordering on some collection of objects.

public interface Comparator<T> {

int compare(T o1, T o2);

}

1. Runnable: The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread.

public interface Runnable {

void run();

}