Java Functional Interfaces Cheat Sheet

Cristian Riță - https://www.linkedin.com/in/cristianrita

Consumer<T>

A Consumer<T> represents an operation that accepts a single argument and returns nothing.
Explanation: *printName* takes the name and print it. Nothing is returned.

```java
public static void main(String[] args) {
    List<String> names = Arrays.asList("Alice", "Bob", "Kevin");
    Consumer<String> printer = name -> System.out.println(name);
    names.forEach(printer);
}
```

Supplier<T>

A Supplier<T> represents a supplier of results.
Explanation: *findUserIdByEmail* randomly returns the id 1 or an empty result. If no user is found *idSupplier* suplies a random integer.

```java
private static Optional<Integer> findUserIdByEmail (String email) {
     if (Math.random() > 0.5) return Optional.of(1);
     return Optional.empty();
}

public static void main(String[] args) {
      Supplier idSupplier = () -> new Random().nextInt(100);
      System.out.println(findUserIdByEmail("test@email.com")
                        .orElseGet(idSupplier));
}
```

Predicate<T>

A Predicate<T> represents a predicate (boolean-valued function) of one argument.
Explanation: *isEven* predicate takes an integer an returns a boolean.

```java
public static void main(String[] args) {
      Predicate isEven = (number) -> number % 2 == 0;
      Stream numbers = Stream.of(1, 2, 3, 4, 5, 6, 7, 8);
      System.out.println(numbers.filter(isEven).count())
}
```

Function<T, R>

A Function<T, R> represents a function that accepts one argument and produces a result.
Explanation: *isEven* predicate takes an integer an returns a boolean.

```java
public static void main(String[] args) {
       Stream numberList = Stream.of(10, 20, 30);
       Function multiplier = (x) -> x * 2;
       numberList.map(multiplier).forEach(System.out::println);
}
```

BinaryOperator<T>

A BinaryOperator<T> represents an operation upon two operands of the same type, producing a result of the same type as the operands.
Explanation: *adder* takes two integeres and returns an integer.

```java
public static void main(String[] args) {
      List<Integer> myList = Arrays.asList(1, 2, 3, 4, 5);
      BinaryOperator<Integer> adder = (acc, x) -> acc + x;
      System.out.println(myList.stream().reduce(0, adder));
}
```