Java 8 Best Practices Cheat Sheet

For more awesome cheat sheets visit rebellabs.org!

REBELLABS by ZEROTURNAROUND

BROUGHT TO YOU BY JRebel

Default methods

Evolve interfaces & create traits

```
//Default methods in interfaces
@FunctionalInterface
interface Utilities {
  default Consumer<Runnable> m() {
    return (r) -> r.run();
  }
  // default methods, still functional
  Object function(Object o);
}
class A implements Utilities { // implement
  public Object function(Object o) {
    return new Object();
  }
  {
    // call a default method
    Consumer<Runnable> n = new A().m();
  }
}
```

Lambdas

Syntax:
(parameters) -> expression
(parameters) -> { statements; }

```
// takes a Long, returns a String
Function<Long, String> f = (l) -> l.toString();
// takes nothing gives you Threads
Supplier<Thread> s = Thread::currentThread;
// takes a string as the parameter
Consumer<String> c = System.out::println;

// use them with streams
new ArrayList<String>().stream().
// peek: debug streams without changes
peek(e -> System.out.println(e)).
// map: convert every element into something
map(e -> e.hashCode()).
// filter: pass some elements through
filter(e -> ((e.hashCode() % 2) == 0)).
// collect all values from the stream
collect(Collectors.toCollection(TreeSet::new))
```

java.util. Optional

A container for possible null values

```
// Create an optional
Optional<String> optional =
Optional.ofNullable(a);
// process the optional
optional.map(s -> "RebelLabs:" + s);
// map a function that returns Optional
optional.flatMap(s -> Optional.ofNullable(s));

// run if the value is there
optional.ifPresent(System.out::println);
// get the value or throw an exception
optional.get();

// return the value or the given value
optional.orElse("Hello world!");
// return empty Optional if not satisfied
optional.filter(s -> s.startsWith("RebelLabs"));
```

Rules of Thumb

Traits: 1 default method per interface
Don't enhance functional interfaces
Only conservative implementations

Expressions over statements
Refactor to use method references
Chain lambdas rather than growing them

Fields - use plain objects
Method parameters, use plain objects
Return values - use Optional
Use *orElse()* instead of *get()*