Guava Reference Card

Functional Units

```
interface Function<F,T>  T apply(F)          F
interface Predicate<T>   boolean apply(T)    P
interface Supplier<T>    T get()             S

Functions   .identity()
            .constant(E)
            .forSupplier(S<T>)
            .forMap(Map<F,T>)

Predicates  .alwaysTrue() / .alwaysFalse()
            .isNull() / .notNull()
            .equalTo(T)
            .instanceOf(Class<?>)
            .in(Collection<? extends T>)
            .not(P<T>)
            .and(P<T>,P<T>) / .or(P<T>,P<T>)

Suppliers   .compose(F<F,T>, S<F>)
            .memoize(S<T>)
```

Functional Operations

```
Iter{able/ator}s
  boolean .all(Iter<T>, P<T>)
  boolean .any(Iter<T>, P<T>)
  Iter<T> .filter(Iter<T>, P<T>)
  T       .find(Iter<T>, P<T>, T default)
  Iter<T> .transform(Iter<F>, F<F,T>)

Collections2
  Coll<E> .filter(Coll<E>, P<E>)
  Coll<T> .transform(Coll<F>, F<F,T>)

Lists
  List<T>       .reverse(List<T>)
  List<List<T>> .partition(List<T>, int size)
  List<T>       .transform(List<F>, F<F,T>)
```

Iterable Builders

```
Immutable{Map/List/Collection/Set}
  .of(...)
  .copyOf({Map/List/Collection/Set})
  .builder()

Iterators
  Iterator<T>    .forArray(T...)
  Iterator<T>    .forEnumeration(Enumeration<T>)
```

```
Lists
  ArrayList<T>   .newArrayList(T...)
  LinkedList<T>  .newLinkedList(T...)

Maps
  HashMap<K,V>   .newHashMap()
  TreeMap<K,V>   .newTreeMap()
  Map<K,V>       .difference(Map<K,V>, Map<K,V>)
  Map<K,V>       .uniqueIndex(Iter<V>, F<V,K>)
  Map<Str,Str>   .fromProperties(Properties)

Sets
  HashSet<E>     .newHashSet()
  TreeSet<E>     .newTreeSet()
  Set<Enum<E>>   .complementOf(Collection<Enum<E>>)
  Set<E>         .newSetFromMap(Map<E, Boolean>)
  Set<E>         .union(Set<E>, Set<E>)
  Set<E>         .intersection(Set<E>, Set<?>)
  Set<E>         .difference(Set<E>, Set<?>)
  Set<E>         .cartesianProduct(List<Set<E>>)

new MapMaker()
  .concurrencyLevel(int)
  .weakKeys() / .weakValues()
  .makeComputingMap(F<K,V>)
```

Caches

```
interface Cache<K,V>  V get(K)
                      V getUnchecked(K)
                      void invalidate(K)
                      void invalidateAll()
                      long size()
                      CacheStats stats()
                      Map<K,V> asMap()
                      void cleanUp()

CacheBuilder.newBuilder()
  .concurrencyLevel(int)
  .weakKeys()
  .maximumSize(int)
  .expireAfterWrite(int, TimeUnit)
  .build(CacheLoader.from(F<K,V>))
  .build(CacheLoader.from(S<V>))
```

Utility

```
Preconditions   .checkArgument(boolean)
                .checkState(boolean)
                .checkNotNull(T)
                .checkElementIndex(int i, int size)

Objects
   boolean      .equal(Object, Object)
   T            .firstNotNull(T, T)
   Helper       .toStringHelper(Class<?>)

Throwables
   Throwable    .getRootCause(Throwable)
   String       .getStackTraceAsString(Throwable)

Strings
   String       .nullToEmpty(String)
   String       .emptyToNull(String)
   boolean      .isNullOrEmpty(String)
   String       .padStart(String, int, char)
   String       .padEnd(String, int, char)
   String       .repeat(String, int)
```

Tokenisation

```
Joiner    .on(String)
          .skipNills()
          .useForNull(String)
          .join(Iterable<?>)

Splitter  .on(String)
          .omitEmptyStrings()
          .trimResults()
          .limit(int)
          .split(String)
```

Ranges

```
class Range<C extends Comparable> {
  boolean  contains(C)
  boolean  containsAll(Iterable<C>)
  boolean  encloses(Range<C>)
  boolean  isConnected(Range<C>)
  Range<C> intersection(Range<C>)
  Range<C> span(Range<C>)
}

Ranges
  Range<C> .open(C,C) / .closed(C,C)
  Range<C> .lessThan(C) / .greaterThan(C)
  Range<C> .atMost(C) / .atLeast(C)
  Range<C> .all()
  Range<C> .singleton(C)
  Range<C> .encloseAll(Iterable<C>)
```