Google's Guava

Robert Bachmann

May 2, 2011

Outline

- Basic utilities
- Collections
- Functions and Predicates
- MapMaker

Example class (1/2)

```
public class Person {
  public final String name;
  public final String email;
  public final Gender gender;

  public Person(String name, String email,
      Gender gender) {
    this.name = name;
    this.email = email;
    this.gender = gender;
  }

  // hashCode, equals
}
```

Example class (2/2)

```
@Override
public int hashCode() {
  return Objects.hashCode(name, gender, email);
}

@Override
public boolean equals(Object obj) {
  if (this == obj) return true;
  if (obj == null || getClass() != obj.getClass())
    return false;
  Person other = (Person) obj;
  return Objects.equal(name, other.name)
      && Objects.equal(gender, other.gender)
      && Objects.equal(email, other.email);
}
```

Basic utilities

```
@Test
public void charsets() {
  assertSame(
      Charset.forName("UTF-8"), // JDK
      Charsets.UTF_8 // Guava
  );
  // same for: UTF16, ISO_8859_1, US_ASCII
}
```

Basic utilities

```
@Test
public void defaults() {
  assertEquals(new Integer(0),
      Defaults.defaultValue(int.class));
  assertEquals(new Double(0),
      Defaults.defaultValue(double.class));
  assertEquals(null,
      Defaults.defaultValue(Double.class));
  assertEquals(null,
      Defaults.defaultValue(String.class));
  assertEquals(null,
      Defaults.defaultValue(Person.class));
}
```

Basic utilities

```
// our test data
final Person alice =
  new Person("Alice Doe", "alice@example.com", Gender.F);
final Person bob =
  new Person("Bob Doe", "bob@example.com", Gender.M);
final Person john =
  new Person("John Doe", "john@example.com", Gender.M);
final Person jane =
  new Person("Jane Doe", "jane@example.com", Gender.F);
```

Basic utilities

```
@Test
public void equal() {
  Person bob2 =
    new Person("Bob Doe", "bob@example.com", Gender.M);
  boolean eq =
    Objects.equal("a", "a") &&
    Objects.equal(bob, bob2) &&
    Objects.equal(null, null);
  assertTrue(eq);
  assertFalse(Objects.equal(null, "a"));
  assertFalse(Objects.equal("a", null));
}
```

Basic utilities

```
@Test
public void firstNonNull() {
  Person p;
  p = Objects.firstNonNull(null, bob);
  assertSame(bob, p);
  p = Objects.firstNonNull(alice, bob);
  assertSame(alice, p);
  p = Objects.firstNonNull(alice, null);
  assertSame(alice, p);
  try {
    Objects.firstNonNull(null, null);
  }
  catch (NullPointerException npe) {return;}
  fail("No exception");
}
```

Basic utilities

```
public String titleForPerson(Person person) {
  Preconditions.checkNotNull(person, "Person is null");
  return Gender.M.equals(person.gender) ?
      "Mr." : "Ms.";
}
@Test
public void preconditons() {
  assertEquals("Ms.", titleForPerson(jane));
  try {
    titleForPerson(null);
  }
  catch (NullPointerException npe) {
    assertEquals("Person is null", npe.getMessage());
  }
}
```

Basic utilities

```
@Test
public void morePreconditons() {
  boolean expression = true;
  Preconditions.checkArgument(expression); // throws IAE
  Preconditions.checkState(expression); // throws ISE
  // also:
  // Preconditions.checkPositionIndex(index, size);
}
```

Basic utilities

```
// com.google.common.base.Equivalence<T>
public abstract interface Equivalence<T> {
  public abstract boolean equivalent(
    @Nullable T paramT1,
    @Nullable T paramT2);
  public abstract int hash(@Nullable T paramT);
}
```

Basic utilities

```
@Test
public void joiner() {
  Joiner joiner = Joiner.on("; ").skipNulls();
  String result = joiner.join('X',1,null,23);
  assertEquals("X; 1; 23", result);
  joiner = Joiner.on(':').useForNull("<missing>");
  result = joiner.join('X',1,null,23);
  assertEquals("X:1:<missing>:23", result);
}
```

Basic utilities

```
@Test
public void splitter() {
  Iterable<String> parts =
    Splitter.on(',').split("foo,,bar, quux");
  String[] array = Iterables.toArray(parts, String.class);
  assertEquals(new String[]{"foo","","bar"," quux"},
      array);

  parts = Splitter.on(',').omitEmptyStrings()
    .split("foo,,bar, quux");
  array = Iterables.toArray(parts, String.class);
  assertEquals(new String[]{"foo","bar"," quux"}, array);
}
```

Basic utilities

```
@Test
public void equivalence() {
  Equivalence<Object> eq = Equivalences.equals();
  Equivalence<Object> same = Equivalences.identity();

  String s1 = "abc";
  String s2 = String.format("ab%s", "c");

  assertTrue(eq.equivalent(s1, s2));
  assertFalse(same.equivalent(s1, s2));
}
```

Basic utilities

```
Equivalence<String> eqIC = new Equivalence<String>() {
  @Override
  public boolean equivalent(String a, String b) {
    return (a == null) ? (b == null)
        : a.equalsIgnoreCase(b);
  }

  @Override
  public int hash(String s) {
    return (s == null) ? 0 : s.toLowerCase().hashCode();
  }
};

@Test
public void customEquivalence() {
  assertTrue(eqIC.equivalent("abc", "ABC"));
}
```

Basic utilities

```
@Test
public void pairwiseEquivalence() {
  List<String> list = new ArrayList<String>();
  list.add("a");
  list.add("b");
  List<String> list2 = new LinkedList<String>();
  list2.add("a");
  list2.add("B");
  boolean eq =
    Equivalences.pairwise(eqIC).equivalent(list, list2);
  assertTrue(eq);
}
```

Collections

- BiMap
- ClassToInstanceMap
- Constraint
- Forwarding Collections
- Multiset
- Multimap
- Interner
- Table

Collections

```
@Test
public void utils() {
  List<Person> persons =
    Lists.newArrayList(alice, bob);
  assertEquals(2, persons.size());
}
```

Collections

```
@Test
public void immutable() {
  ImmutableList<Person> persons =
    ImmutableList.of(alice,bob);
  assertEquals(2, persons.size());
  try {
    persons.add(john);
  }
  catch(UnsupportedOperationException uoe) {
    return;
  }
  fail("No exception");
}
```

Functions and Predicates

```
Function<Person, String> titleFunc =
  new Function<Person, String>() {
    @Override
    public String apply(Person p) {
      return titleForPerson(p) + " " + p.name;
    }
};
```

Functions and Predicates

```
@Test
public void apply() {
  ImmutableList<Person> persons =
    ImmutableList.of(alice,bob);
  List<String> names =
    Lists.transform(persons, titleFunc);
  assertEquals("Ms. Alice Doe", names.get(0));
  assertEquals("Mr. Bob Doe", names.get(1));
  assertEquals(2, names.size());
}
```

Functions and Predicates

```
@Test
public void filter() {
  ImmutableList<Person> persons =
    ImmutableList.of(alice,bob);
  Iterable<Person> females =
  Iterables.filter(persons, new Predicate<Person>() {
    @Override
    public boolean apply(Person input) {
      return Gender.F.equals(input.gender);
    }
  });
  for (Person p : females) {
    assertSame(p, alice);
  }
}
```

MapMaker

```
ConcurrentMap<Key, Graph> graphs = new MapMaker()
    .expireAfterWrite(10, TimeUnit.MINUTES)
    .makeComputingMap(
        new Function<Key, Graph>() {
          public Graph apply(Key key) {
            return createExpensiveGraph(key);
          }
        });
```