This Github search (@interface) gives you the list of all the annotations :

https://github.com/junit-team/junit/search?q=%22%40interface%22&type=Code

Basic Annotations

@Test @Before @After @AfterClass @BeforeClass @Ignore @Runwith

Parameterized Tests

For Parameterized tests use @Parameters and @RunWith(Parameterized.class)
https://github.com/junit-team/junit/wiki/Parameterized-tests

Category

@Category
Grouping tests into categories. e.g. Fast, Slow etc.

https://github.com/junit-team/junit/wiki/Categories

@IncludeCategory
Runs only the classes and methods that are annotated with either the category given with the @IncludeCategory annotation, or a subtype of that category.

@ExcludeCategory
Inverse of @IncludeCategory

Rules

@Rule
Rules allow very flexible addition or redefinition of the behavior of each test method in a test class. e.g. Creating a Temp Folder rule for creating a temp folder while running tests.

https://github.com/junit-team/junit/wiki/Rules

Theory and related annotations

@Theory
Theories give more flexible and expressive assertions

https://github.com/junit-team/junit/wiki/Theories

@DataPoint
Annotating an field or method with @DataPoint will cause the field value or the value returned by the method to be used as a potential parameter for theories in that class

@DataPoints

Extension of @Datapoint
Annotating an array or iterable-typed field or method with @DataPoints will cause the values in the array or iterable given to be used as potential parameters for theories in that class

@FromDataPoints

Annotating a parameter of a @Theory method with @FromDataPoints will limit the datapoints considered as potential values for that parameter to just the @DataPoints with the given name

@ParametersSuppliedBy
Annotating a @Theory method parameter with @ParametersSuppliedBy causes it to be supplied with values from the named ParameterSupplier when run as a theory

@TestedOn

The @TestedOn annotation takes an array of values to be used as data points for the annotated parameter.

e.g.

```
@Theory

public void multiplyIsInverseOfDivideWithInlineDataPoints(

        @TestedOn(ints = {0, 5, 10}) int amount,

        @TestedOn(ints = {0, 1, 2}) int m

) {

    assumeThat(m, not(0));

    assertThat(new Dollar(amount).times(m).divideBy(m).getAmount(), is(amount));

}
```