Programming Model

new ideas about writing tests

Test Definition

- Keep the old way: @Test, @Before, …
- Enable new ways, e.g.
  - using lambdas
  - support hierarchical/nested structures/contexts, e.g. HierarchicalContextRunner

Dynamic test registration

- required when not all tests are not known before being executed, e.g. nested contexts using lambdas

- support Java 8 Streams of test parameters, e.g. when reading test parameters from an external file or a database

- enable dynamic search for test parameters, e.g. for property-based testing like Quickcheck

Exception Testing using Lambdas

```
assertThrows(IllegalArgumentException.class, () -> {
    doSomething();
});

IllegalArgumentException expected =
expectThrows(IllegalArgumentException.class, () -> {
    doSomething();
});
assertThat(excepted).hasMessage("bla");
```

Interaction with Extensions

- @Rule and @ClassRule are not powerful enough, e.g. SpringClassRule
- Runner API (@RunWith) is not composable: you can only have one runner
- Goal: Separate extensions for separate responsibilities, e.g. test discovery, test parameterization, …
- Instance variables vs. class-level annotations?

Aggregated Assertions

```
Person person = new Person("Johannes", "Link", "Germany");

assertAll(
    () -> assertEquals("Johannes", person.getFirstName()),
    () -> assertEquals("Link", person.getLastName()),
    () -> assertEquals("Germany", person.getCountry())
);
```

Conditional Test Execution

- Ex: @Category, @Ignore, assumptions
- programmatic support:
  - `assumeThat(...), assumeTrue(...)`

  -
    ```
    assumingThat(condition)
    .thenExecute(() -> {
        doSomething();
    })
    ```

More ideas

- Lazy evaluation of failure messages
- method parameter resolution/injection
- Explicit test names: `@Test("test name“)`