

Deep Space Orbiter

odyssey in code

Fabian

Idea

- Implement something with interwoven / complicated “business logic”
- Try Solo development
- Experience evolving requirements
- Experience “the obvious”

And

- Share my experience



So it had to be complicated

```
16  /**
17   * 2D Space for simplicity
18   * <p>
19   * Swiss Space Force launches an Orbiter towards Mars
20   * <p>
21   * The Orbiter starts with its camera point to BACKWARD and its antenna FORWARD
22   * <p>
23   * The antenna and camera can not point in the same direction
24   * The camera can only record if the antenna faces a planet
25   * The camera should record the closest planet as long as possible
26   * (thus the camera should record Earth and after half way record Mars)
27   * The camera can record 2x1 Mkm worth of flight and when it's full it has to transmit
28   * (the camera recording can not be interrupted, it will always record 1M km worth of footage)
29   * The antenna should try to transmit if it's facing earth if the camera has recorded something (and the camera is not recording)
30   * The camera should no longer record if it is full and should instead turn away for the antenna to transmit.
31   * with 100mW
32   * and after 20 km with 200mW
33   * <p>
34   * <p>
35   * The navigationssystem will tell the orbiter current time and distance from the planets.
36   * <p>
37   *
38   *      [      ]
39   * o      BACKWARD [ Orbiter ] FORWARD      0
40   * Earth      [      ]      Mars
41   * <p>
42   * <-----395.04 million km ----->
43   * <p>
44   * For simplificaiton distance is equal to time: Every Mkm = 1h recording
45   * The orbiter can always receive/communicate, the antenna is only required for transmitting "HD Video Recordings".
46   */
```

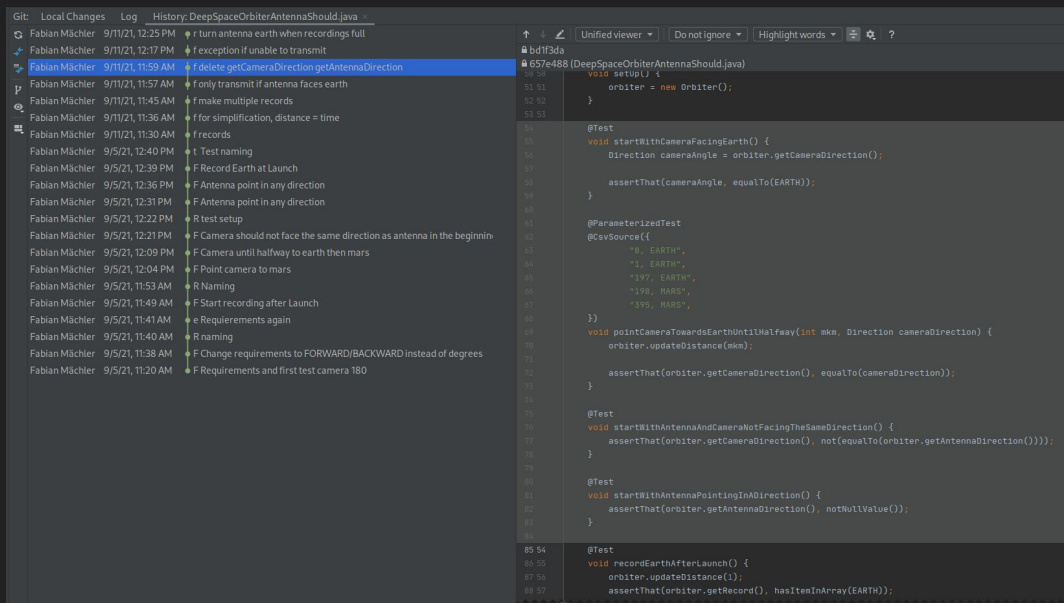
Call Getters in Tests? -> Smell?

- getCameraDirection() and getAntennaDirection() in Testcode
- “Forward-” and “Backward Facing” are internal concepts
- So is Direction Earth and Mars
- Find another way to test it
 - We can observe the received recordings instead
- -> Write new tests

```
@Test
void recordEarthAfterLaunch() {
    orbiter.updateDistance( newDistance: 1);
    assertThat(orbiter.getRecord(), hasItemInArray(EARTH));
}
```

And the nice part

- Delete a lot of old code



```
Git: Local Changes Log History: DeepSpaceOrbiterAntennaShould.java
Fabian Mächler 9/11/21, 12:25 PM r turn antenna earth when recordings full
Fabian Mächler 9/11/21, 12:17 PM f exception if unable to transmit
Fabian Mächler 9/11/21, 11:59 AM f delete getCameraDirection getAntennaDirection
Fabian Mächler 9/11/21, 11:57 AM f only transmit if antenna faces earth
Fabian Mächler 9/11/21, 11:45 AM f make multiple records
Fabian Mächler 9/11/21, 11:36 AM f for simplification, distance = time
Fabian Mächler 9/11/21, 11:30 AM f records
Fabian Mächler 9/5/21, 12:40 PM l Test naming
Fabian Mächler 9/5/21, 12:39 PM f Record Earth at Launch
Fabian Mächler 9/5/21, 12:36 PM f Antenna point in any direction
Fabian Mächler 9/5/21, 12:31 PM f Antenna point in any direction
Fabian Mächler 9/5/21, 12:22 PM r Test setup
Fabian Mächler 9/5/21, 12:21 PM f Camera should not face the same direction as antenna in the beginnin
Fabian Mächler 9/5/21, 12:09 PM f Camera until halfway to earth then mars
Fabian Mächler 9/5/21, 12:04 PM f Point camera to mars
Fabian Mächler 9/5/21, 11:53 AM r Naming
Fabian Mächler 9/5/21, 11:49 AM f Start recording after Launch
Fabian Mächler 9/5/21, 11:41 AM e Requirements again
Fabian Mächler 9/5/21, 11:40 AM r naming
Fabian Mächler 9/5/21, 11:38 AM f Change requirements to FORWARD/BACKWARD instead of degrees
Fabian Mächler 9/5/21, 11:20 AM f Requirements and first test camera 180

657e488 (DeepSpaceOrbiterAntennaShould.java)
29:26 void setup() {
31:51 orbiter = new Orbiter();
32:52 }
33:53
34:
35: @Test
36: void startWithCameraFacingEarth() {
37: Direction cameraAngle = orbiter.getCameraDirection();
38:
39: assertEquals(cameraAngle, equalTo(EARTH));
40: }
41:
42: @ParameterizedTest
43: @CsvSource({
44: "0, EARTH",
45: "90, EARTH",
46: "180, EARTH",
47: "270, MARS",
48: "360, MARS",
49: })
50: void pointCameraTowardsEarthUntilHalfway(int kms, Direction cameraDirection) {
51: orbiter.updateDistance(kms);
52:
53: assertEquals(orbiter.getCameraDirection(), equalTo(cameraDirection));
54: }
55:
56: @Test
57: void startWithAntennaAndCameraNotFacingTheSameDirection() {
58: assertEquals(orbiter.getCameraDirection(), not(equalTo(orbiter.getAntennaDirection())));
59: }
60:
61: @Test
62: void startWithAntennaPointingInADirection() {
63: assertEquals(orbiter.getAntennaDirection(), notNullValue());
64: }
65:
66:
67: @Test
68: void recordEarthAfterLaunch() {
69: orbiter.updateDistance();
70:
71: assertEquals(orbiter.getRecord(), hasItemInArray(EARTH));
72: }
```

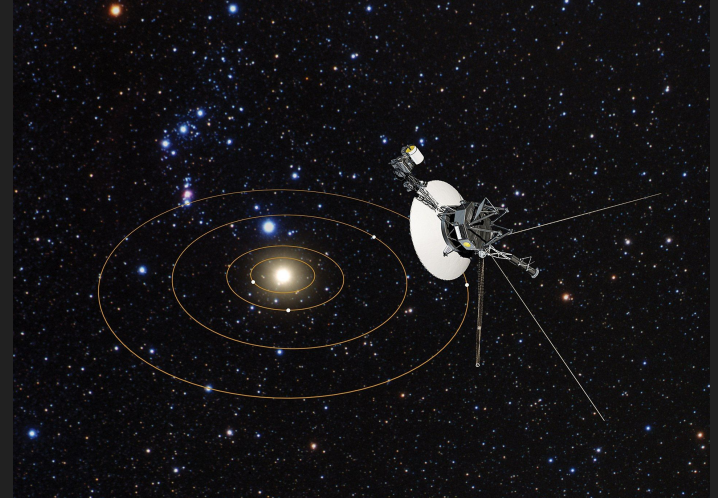
Drastic Implementation change without changes in the Tests

- I could change from “CameraDirection and AntennaDirection” to “Slots which a Devices occupies” without changes in the tests

<pre>public class Orbiter { public static final int DISTANCE_TO_MARS = 395; private int distance; private Direction cameraDirection; private Direction antennaDirection; private final LinkedList<Direction> records = new LinkedList<Direction>(); public Orbiter() { cameraDirection = EARTH; } }</pre>	<pre>8 9 9 10 10 11 11 12 12 13 13 14 14 15 15 16 16 17 17 18 18 19 19 20 20 21 21 22 22 23</pre>	<pre>public class Orbiter { public static final int DISTANCE_TO_MARS = 395; private Device directionEarthSlot; private int distance; private Direction cameraDirection; private final LinkedList<Direction> records = new LinkedList<Direction>(); public Orbiter() { directionEarthSlot = CAMERA; } }</pre>
---	---	--

State of the Project

- About half of the requirements are implemented
- There are refactorings in the production code open
- But the Tests/Requirements seem to be in good shape to me



Other observations

- First degree of freedom: Focus on camera direction, not the recording
 - But it's connected (See Getter-Smells)
- Shaping the requirements involved a lot erroring and thus learning
 - Did I make myself to be the only expert in this field?
- Lot of naming is wrong (I didn't notice)
 - Maybe the Mob would help here too
- Change of requirements during implementation
 - Implementing challenges the requirements
- DirectionEartSlot -> I have nothing telling me how to implement this
 - Try and error?
 - Train the production code like a neural network?

Bonus Slide: Questions along the way

- How do I test the camera and antenna not facing the same way?
 - Expect an Exception if they collide?
 - This could happen in any case
 - Add Assertion in each Test? (they should only test one thing)
- Exceptions to the rescue?

```
@Test
void shouldThrowException_IfTryingToTransmitAndUnableTo() {
    orbiter.updateDistance( newDistance: 1);
    assertThatThrownBy(() -> orbiter.getRecord())
        .isInstanceOf(Exception.class)
        .hasMessageContaining("Antenna faces the wrong way");
}
```

Unfixable Tests?

After implementing the Exception (Antenna does not face Earth now) the following test fails:

```
@Test
void recordEarthAfterLaunch() {
    orbiter.updateDistance( newDistance: 1);
    assertThat(orbiter.getRecord(), hasItemInArray(EARTH));
}
```

But we have new Tests covering this too, just much more complicated

```
@Test
void shouldThrowException_IfTryingToTransmitAndUnableTo() {
    orbiter.updateDistance( newDistance: 1);
    assertThatThrownBy(() -> orbiter.getRecord())
        .isInstanceOf(Exception.class)
        .hasMessageContaining("Antenna faces the wrong way");
}

@Test
void transmitTwoRecordsAfter2mkm() {
    orbiter.updateDistance( newDistance: 1);
    orbiter.updateDistance( newDistance: 2);
    assertThat(orbiter.getRecord(), is(new Direction[]{EARTH, EARTH}));
}
```

-> Interwoven and complicated requirements lead to complicated tests...

Code

You can find the code here: <https://github.com/unSinn/alcor-presentation>