Port me if you can Using ports & adapters



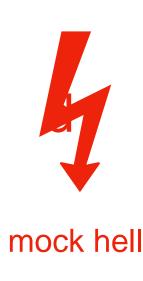
Problem

class EreignisprotokollServiceOldTest

```
Stee
```

```
veid AuslankungSaberstauertEreignis() (
  final Vorgangetracker vorgangetracker - mock(Vorgangetracker, class);
  final Processinformation processinformation = mock(Processinformation.class);
  final SharkDataproviderContext dataproviderContext = mock(SharkDataproviderContext.class, RETURNS DEEP STURNS;
  final BOFactory boFactory = mock(BOFactory.class);
  final SharkSCManager sharkSCManager = mock(SharkSCManager,class);
  final ProcessinformationBO processinformationBO = mock(ProcessinformationBO.class);
  final VorgangBO vorgangBO = mock(VorgangBO,class);
  when (vorgangetracker.getProsessinformation()), thenNeturn (processinformation);
  when (dataproviderContext.getBOFactory()), thenReturn (boFactory);
  when (bofactory.new80Instance (Ereignisprotokoll.class)).thenNeturn (mock (Ereignisprotokoll.class));
  when (dataproviderContext.getSDs()).thenSeturn(sharkSDManager);
  when (sharkSCManager, getProsessinformationSC()), thenReturn (prosessinformationSC);
  when (sharkSCManager, getVorgangSC()), thenReturn (vorgangSC);
  when (vorgangEO.loadVorgangstrackerNonNull(any())).thenNeturn(vorgangstracker);
  when (dataproviderContext.getBOFactory().newBOInstance(AusfuehrungsEintragRef.class)), thenReturn(mock(AusfuehrungsEintragRef.class));
  when (dataproviderContext.getPersistenceManager().getObjectByld(eg(AusfuehrungsEintrag.class), any())).thenReturn(
     mock(husfushrungsEintrag,class));
```

```
final ArgumentCaptor@retration():addEretration() - ArgumentCaptor.forClassChreighisprotokoll.class()
verify(vorgangstracker.getPropessinformation()).addEretration();addEretration();
final Eretration():artEretration():artEretration();
verify(eretration():artEretration();artEretration());
verify(eretration():artEretration());artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretration();artEretra
```



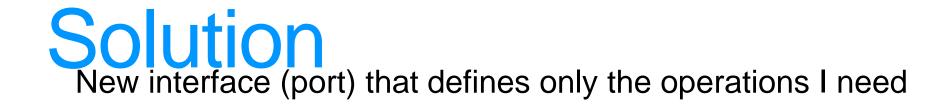
Problem



- dataproviderContext is our class (but in a different modul)
- dataproviderContext provides many functions
- has many nested objects
- I need just few operations, but on different nested objects
- I have to know many internals on how dataproviderContext is used
- how can I reduce/simplify the mocking stuff?



treat dataproviderContext as external and use ports/adapters to decouple



twid addirecprisprotokell(direcprisprotokell) erecprisprotokell);

Erecprisprotokell(newErecprisprotokell)Instance();

AccordantrumpeEintrupheE newEconfuntrumpeEintrupheEinstance();

AccordantrumpeEintrup proEconfuntrumperintrupheEintrupheEinstance();

(ptional=Echecitanchritt> proEktowilstanchrittschritt(Forgangetrucherlid vorgangetrucherlid);

The implementation (adapter) just takes the original DataproviderContext as a constructor parameter and delegates all methods to it.

The EreignisprotokollService gets the new Dataprovider interface injected (instead the old DataproviderContext)

```
dataprovider = mock(Dataprovider.class);
ereignisprotokollService = new EreignisprotokollServiceNewImpl(dataprovider);
```

Solution

```
- Lana EreignisprotokollServiceNewShould {
  private Dataprovider dataprovider = mock(Dataprovider. :lank);
  private AuslenkungUebersteuertEreignis egeignis;
  private EreignisprotokollServiceNew ereignisprotokollService;
  BleforeEach
  void setUp() (
     -dataprovider = mock(Dataprovider, class);
     ereignisprotokoilService = new EreignisprotokollServiceNewImpl(dataprovider);
     ereignin = nuli;
  word annienkungflebersteuertEreighingrotckollieren() (
     ereignisTrittAuf();
     undWirdProtokolliert();
     dannWirdEsPersistiert();
   private void creiquisTrittAuf() (
     List<UebersteuerterFehler> uebersteuerterFehler = Arrays.asList(
        -pew UebersteuerterFehler(new Ausfuehrungseintrag[dentity("fehler 1"), new RuleNusber("rule-1")),
        new UebersteuerterFehler(new AusfuehrungseintragTdentity("teller:2"), new RuleNumber(":ule 2"))
     7.2
     ereignis = AuslenkungUebersteuertEreignis.builder()
        .vorgangstrackerId(new VorgangstrackerId("vorgangstracker id"))
        .gui(StepKeyEnum.HK150 BEHANDLUNGSGRUND)
        .uebersteuerteFehler(uebersteuerterFehler)
  private void undWirdPretokolliert() /
     when (dataprevider.newEreignisprotokollInstance()).thenReturn(mock(Ereignisprotokoll.class));
     when (dataprovider, newAusfuehrungsEintragRefInstance()).thenReturn (mock(AusfuehrungsEintragRef.class));
     ereignisprotokollService.protokolliereAuslenkungUebersteuertEreignis(ereignis);
  private void dannWirdEnPersistiert() (
     final ArgumentCaptor<Ereignisprotokoll> captureEreignisprotokoll ~ ArgumentCaptor.forClass(Ereignisprotokoll.class);
      verify(dataprovider).addEreignisprotokoll(captureEreignisprotokoll.capture());
     final Ereignisprotokoll ereignisprotokoll = captureEreignisprotokoll.getValue();
     verify(ereignisprotokoll).setEreignis(Ereignistyp.AUSLENEUNG UEBERSTEUERT.name());
      verify(ereignisprotokoll).setEreigniszeitpunkt(any());
      verify(ereignisprotokoll).setGui(StepKeyEnum.HK2*0 BEHANDLUNKESGRUND.getName());
      verify(ereignisprotokoll).setAusloesenderUser("TO shark?");
     ArgumentCaptor - AusfuehrungsEintragRef - capturedAusfuehrungsEintrafRef - ArgumentCaptor . forClass (AusfuehrungsEintragRef . class) /
      verify(ereignisprotokoll, times()).addAusfuehrungsEintragkef(capturedAusfuehrungsEintrafRef.capture());
     List<AusfuehrungsEintragRef> erwarteteAusfuehrungsEintrafRef = capturedAusfuehrungsEintrafRef.getAllValues();
      assertThat(erwarteteAusfuehrungsEintrafRef, hasSize(P));
      verily(erwarteteAusfuchrungsEintrafRef.get(0)).setActingRuleNumber("role 1");
     verify(erwarteteAusfuehrungsEintrafRef.get(1)).setActingRuleNumber("rule-2");
     verifyNoMoreInteractions(ereignisprotokoll);
```

mocking hell reduced to these lines

- Conclusion
 ports/adapters can also be used for decoupling internal services (even if it's not a proper ports/adapters implementation, it was he
 - has reduced the complexity in the current implementation
 - testing was much easier
 - allowed me to define convenience methods to access legacy code

Merçi for listening

and thank you for this great course

Contact:

Mail: juerg.weilenmann@css.ch

Twitter: -

Facebook: -

LinkedIn: -

Instagram: -

WhatsApp: -

Git: -

BeachBar: 18:00 - 20:00