Going a little further with Stockportfolio project

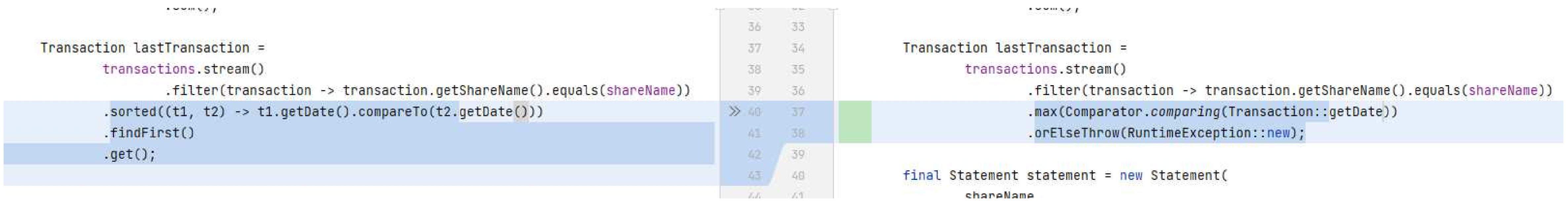
25. May 2021

Urs Birrer

Content

Step 1: Fixing last issue

Remember, when time run out…



```
    Transaction lastTransaction =
            transactions.stream()
                    .filter(transaction -> transaction.getShareName().equals(shareName))
            .sorted((t1, t2) -> t1.getDate().compareTo(t2.getDate()))
            .findFirst()
            .get();
```

```
    Transaction lastTransaction =
            transactions.stream()
                    .filter(transaction -> transaction.getShareName().equals(shareName))
                    .max(Comparator.comparing(Transaction::getDate))
                    .orElseThrow(RuntimeException::new);

    final Statement statement = new Statement(
            shareName
```

Step 2: Exchange repository fake to mockito solution

repository fake or mockito solution

SPOILER!!!

exchange repository fake to mockito solution

- Mixing
  - `ArgumentCaptor.getAllValues()`
- and
  - `when().doReturn()`
- Bad idea
- Not able to get it to work ☹

- https://medium.com/@CDehning/when-to-use-fakes-instead-of-mocks-c80188b9a3f1

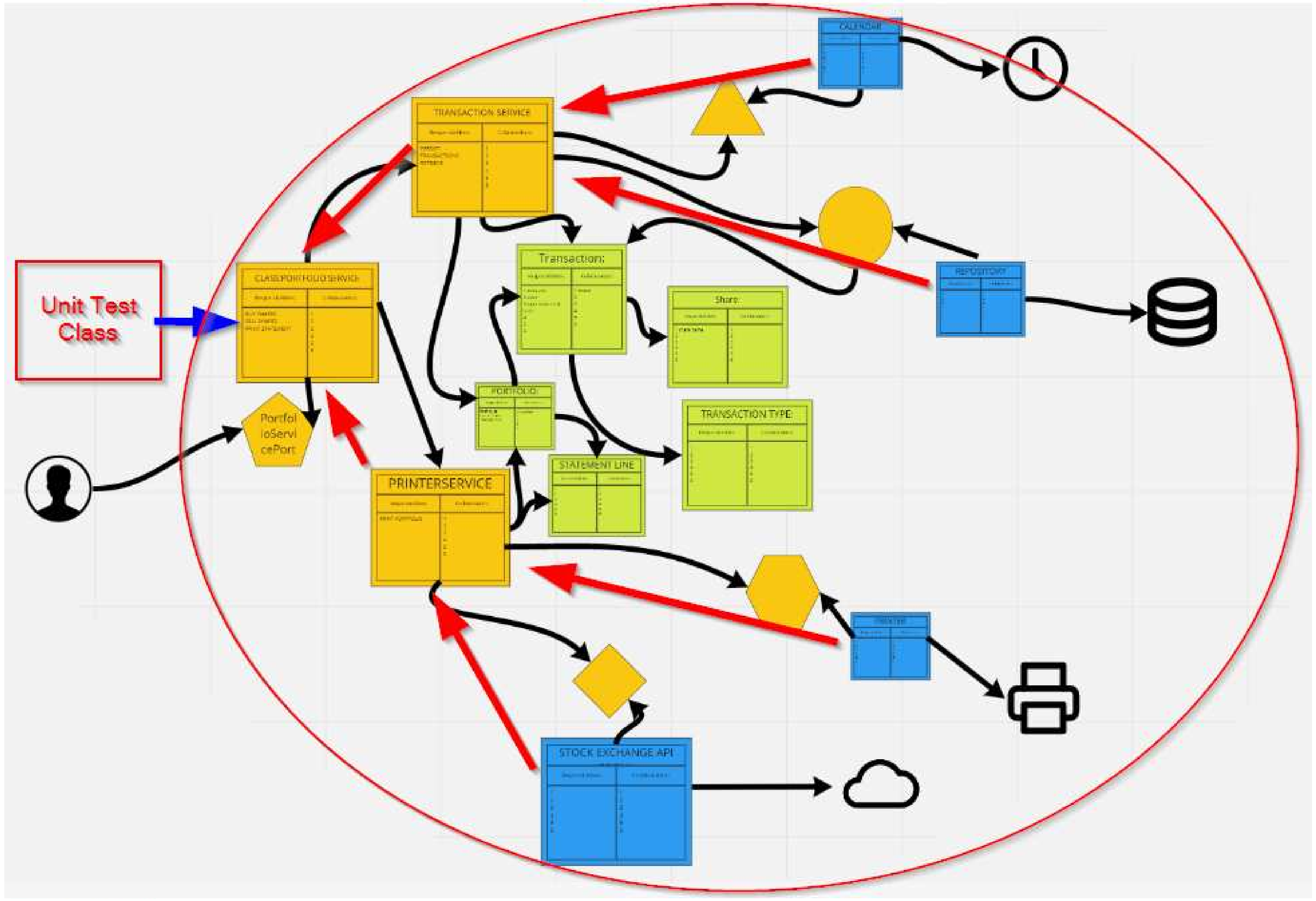
Step 3: Switch implementation to a managed bean version

Switch implementation to a managed bean version

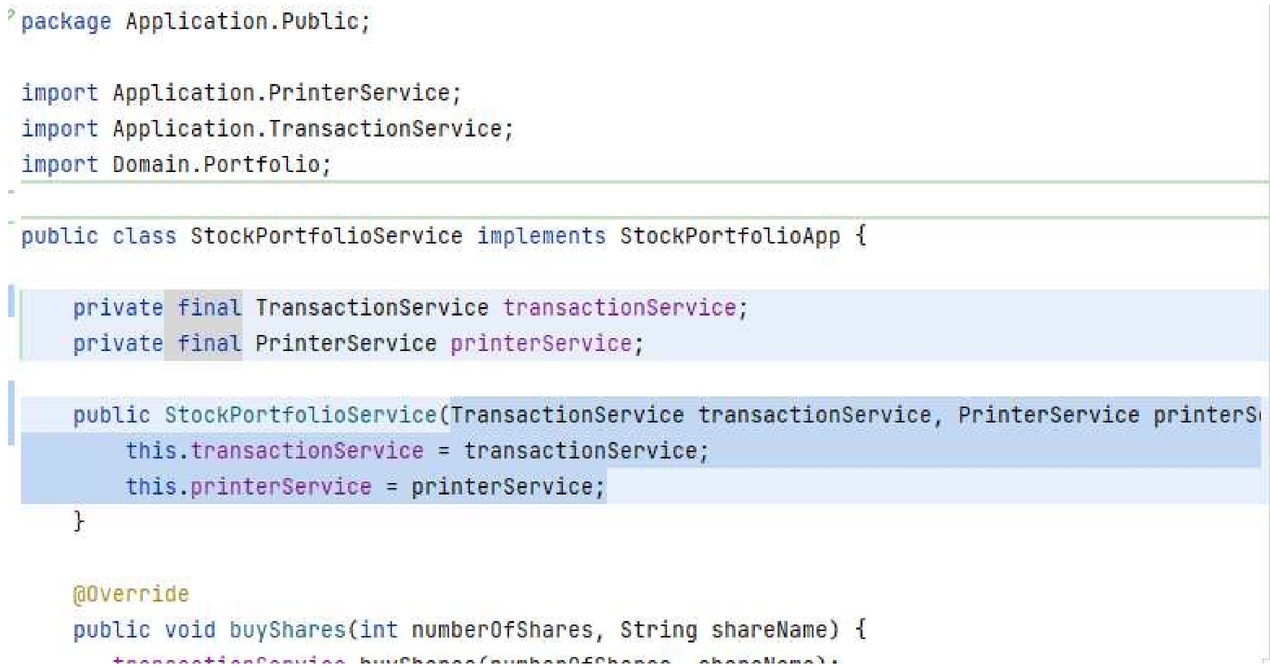
- Possible reasons
  - What if stubs and mock create endless constructors?
  - What if environment works with managed bean?

- Let’s give it a try… Using Cdi, Weld

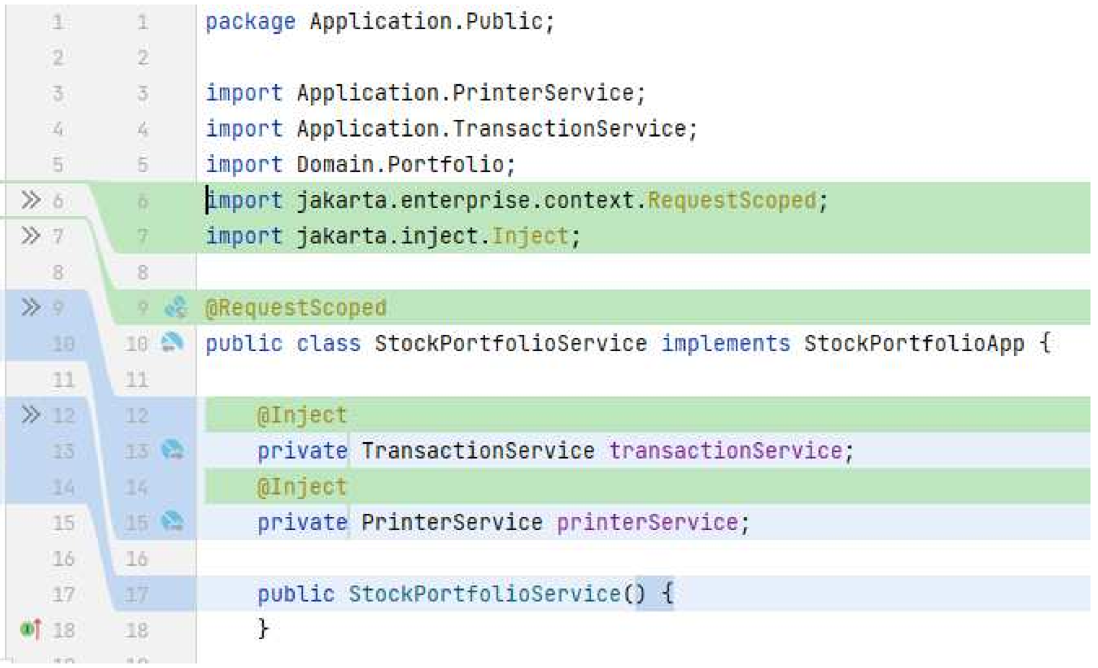
Remember, when there was a design…



Constructor → Inject

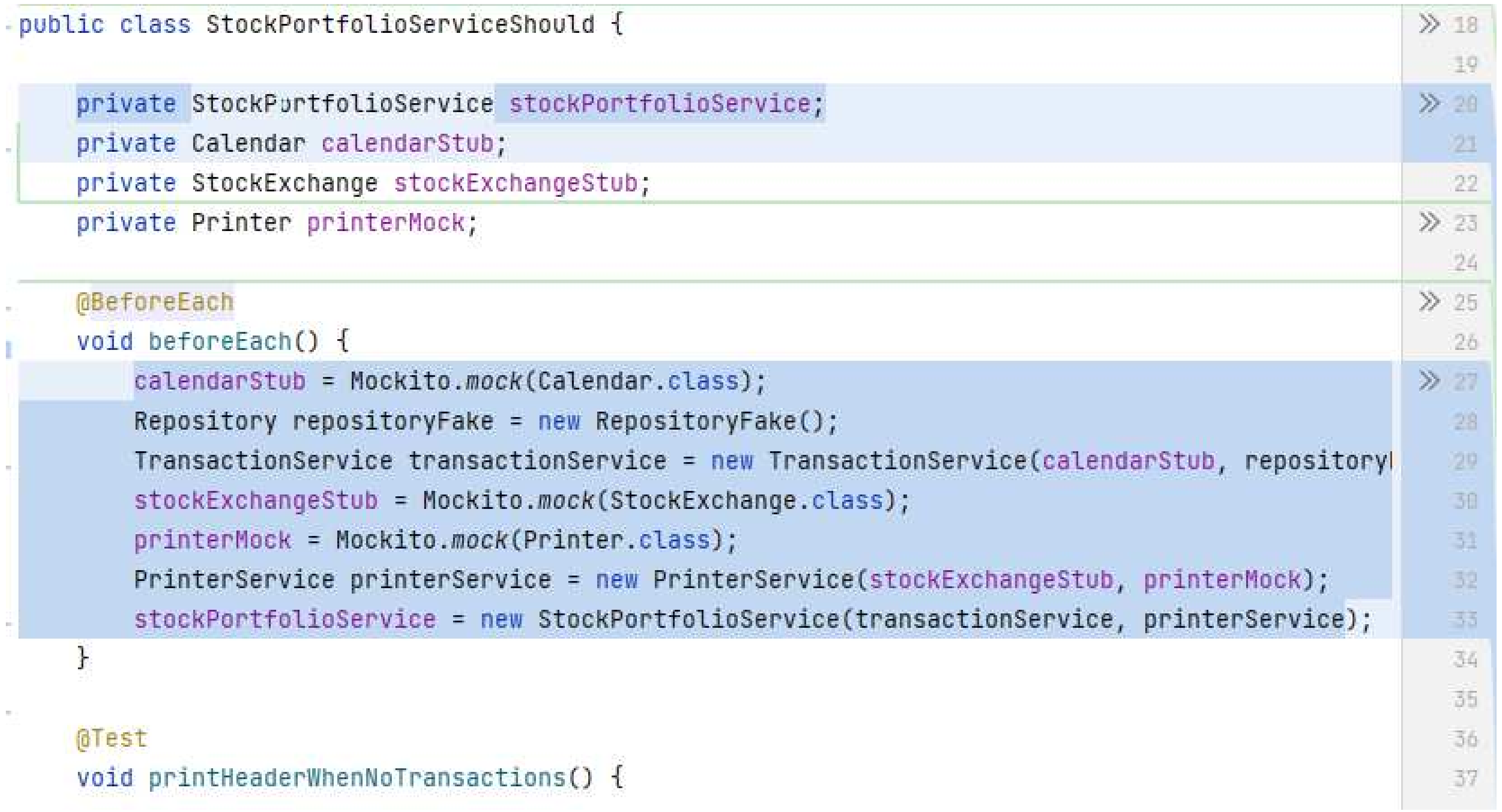


```
package Application.Public;

import Application.PrinterService;
import Application.TransactionService;
import Domain.Portfolio;

public class StockPortfolioService implements StockPortfolioApp {

    private final TransactionService transactionService;
    private final PrinterService printerService;

    public StockPortfolioService(TransactionService transactionService, PrinterService printerS
        this.transactionService = transactionService;
        this.printerService = printerService;
    }

    @Override
    public void buyShares(int numberOfShares, String shareName) {
        transactionService.buyShares(numberOfShares, shareName);
```



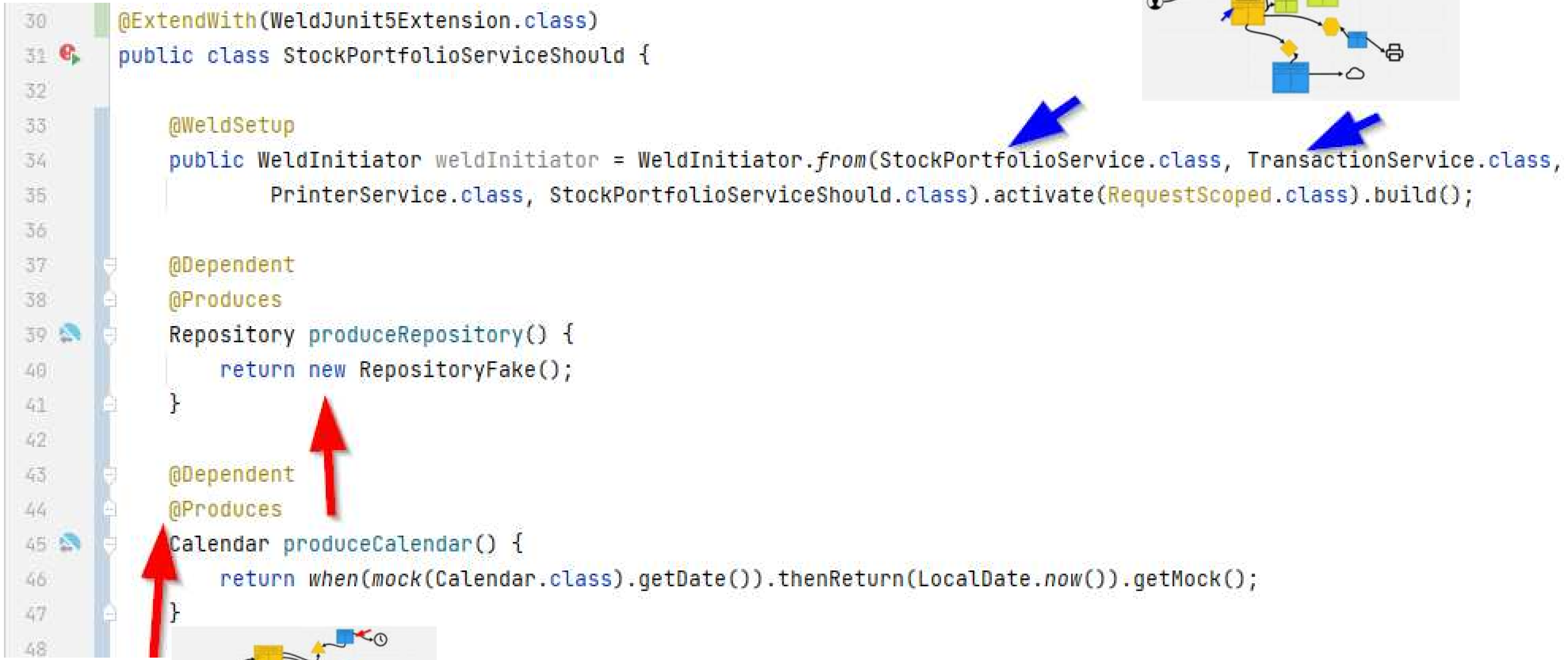
```
package Application.Public;

import Application.PrinterService;
import Application.TransactionService;
import Domain.Portfolio;
import jakarta.enterprise.context.RequestScoped;
import jakarta.inject.Inject;

@RequestScoped
public class StockPortfolioService implements StockPortfolioApp {

    @Inject
    private TransactionService transactionService;
    @Inject
    private PrinterService printerService;

    public StockPortfolioService() {
    }
```

Test class before

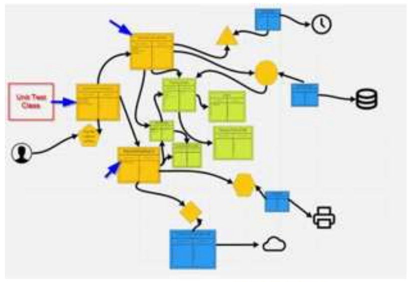


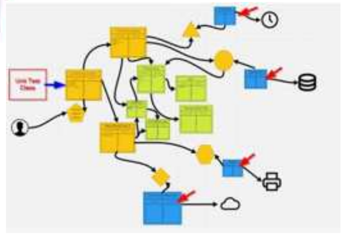
```
public class StockPortfolioServiceShould {

    private StockPortfolioService stockPortfolioService;
    private Calendar calendarStub;
    private StockExchange stockExchangeStub;
    private Printer printerMock;

    @BeforeEach
    void beforeEach() {
        calendarStub = Mockito.mock(Calendar.class);
        Repository repositoryFake = new RepositoryFake();
        TransactionService transactionService = new TransactionService(calendarStub, repository
        stockExchangeStub = Mockito.mock(StockExchange.class);
        printerMock = Mockito.mock(Printer.class);
        PrinterService printerService = new PrinterService(stockExchangeStub, printerMock);
        stockPortfolioService = new StockPortfolioService(transactionService, printerService);
    }

    @Test
    void printHeaderWhenNoTransactions() {
```

Test class after



```
@ExtendWith(WeldJunit5Extension.class)
public class StockPortfolioServiceShould {

    @WeldSetup
    public WeldInitiator weldInitiator = WeldInitiator.from(StockPortfolioService.class, TransactionService.class,
            PrinterService.class, StockPortfolioServiceShould.class).activate(RequestScoped.class).build();

    @Dependent
    @Produces
    Repository produceRepository() {
        return new RepositoryFake();
    }

    @Dependent
    @Produces
    Calendar produceCalendar() {
        return when(mock(Calendar.class).getDate()).thenReturn(LocalDate.now()).getMock();
    }
```





State of work

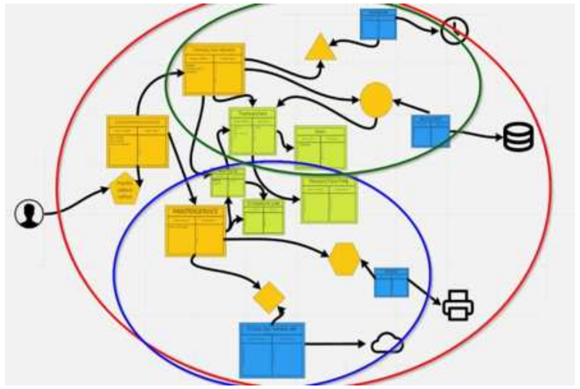
- OK
  - Container starts up
  - Beans are recognized
  - Transaction is stored in Repository

But, the time…

- NOK
  - Stub for stockexchange

Conclusion / Learning

- Fake are pretty easy to implement and should be used
- Using BeanContainer in Unit tests are difficult to ramp up
- Instead it would be easier to focus on unit tests
  - What is the focus of the test?
  - What should be testet?



Thank you!

Literature & References

- https://medium.com/@CDehning/when-to-use-fakes-instead-of-mocks-c80188b9a3f1
- http://weld.cdi-spec.org/
- https://weld.cdi-spec.org/news/2017/12/19/weld-meets-junit5/
- https://github.com/weld/weld-junit
- https://dzone.com/articles/weld-junit-easy-testing-of-cdi-beans
- https://www.investmentexecutive.com/wp-content/uploads/sites/3/2018/04/800x600-businessman-fork-road-career-alphaspirit-20178074.jpg