

Vika 10: Þróun með prófun

- Miðvikudagur 12.3
 - Þróun með prófun (8. kafli)
 - Tímaverkefni



Test-driven
development

Þróun með prófun

- Aðferðafræði þar sem prófanir stýra ferðinni

1. Skrifum **fyrst** einingapróf

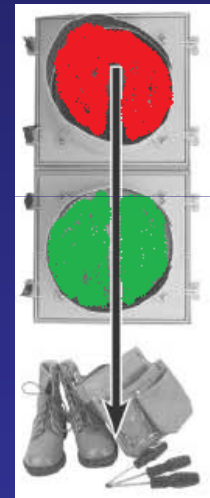
- sem eiga að feila (sbr. rautt ljós)

1. Skrifum kóða sem stenst prófið

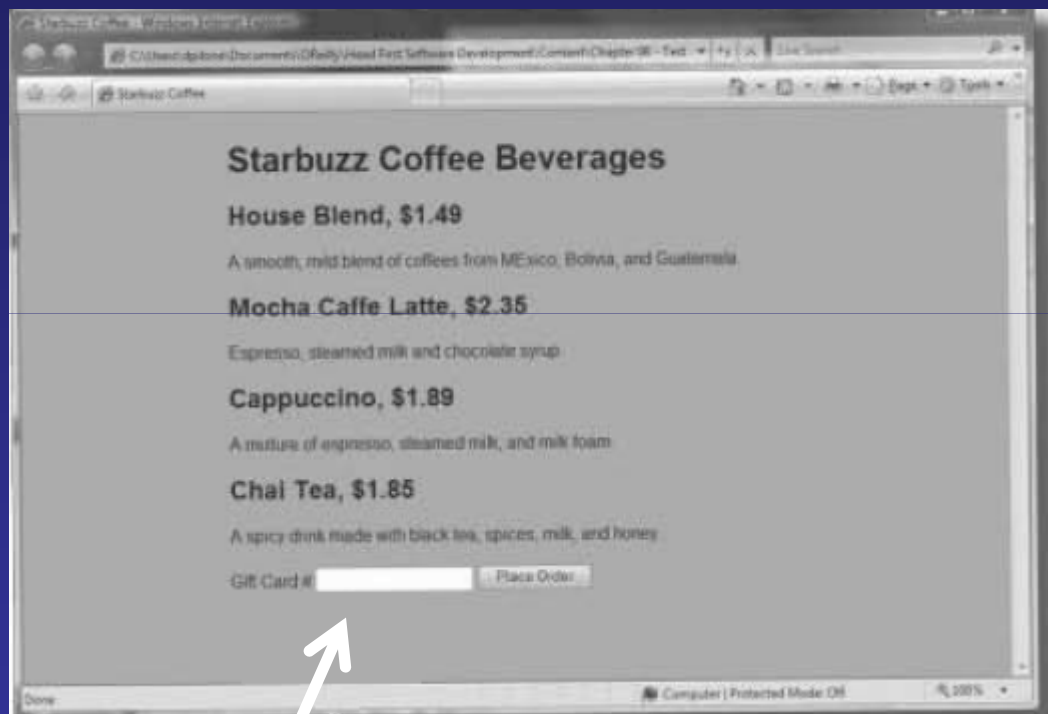
- Eins einfaldan og mögulegt er
- Nú eigum við að fá grænt á einingaprófið.

2. Endurskrifum/lögum til kóða

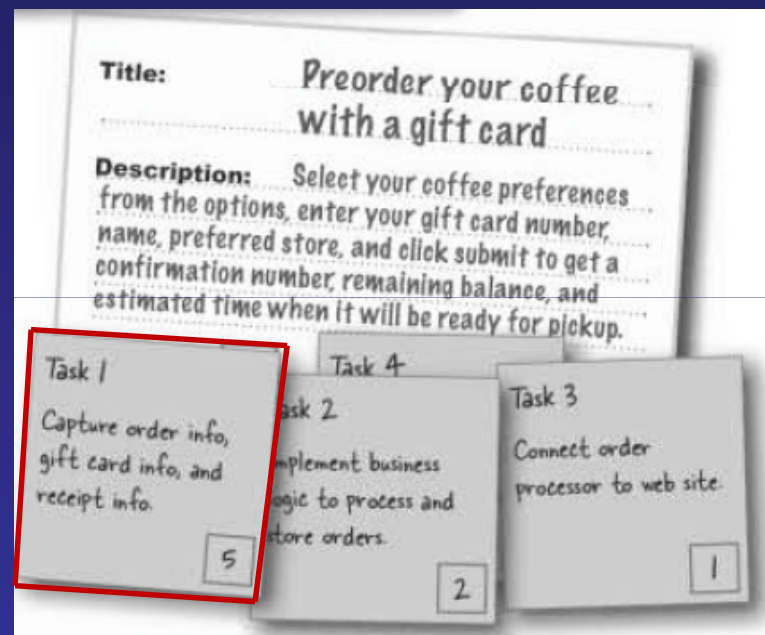
(refactor)



Dæmi: Gjafakort á Starbuzz

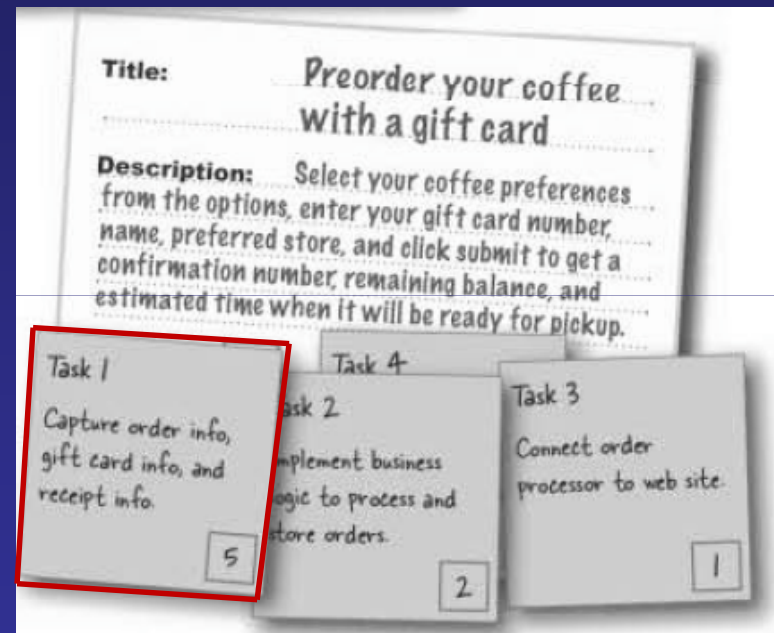


Viðskiptavinur Starbuzz
slær inn kóða á gjafakorti



Dæmi: Fyrsti verkþáttur

- Halda utan um
 - i. Pöntun: Nafn viðskiptavinar, nafn á drykk, ...
 - ii. Gjafakort: Gildistíma korts (upphaf og endi), inneign á korti
 - iii. Kvittun: Inneign, hvenær pöntun v. sótt




Er í raun 3 litlir verkþættir sem við sameinum í einn

Fyrsta einingaprófið

- Skrifum einfaldasta mögulega próf

```
class TestOrderInformation(unittest.TestCase):  
    def test_create_order_information(self):  
        # Test the creation of an object  
        order_info = OrderInformation()  
  
if __name__ == '__main__':  
    unittest.main(exit=False)
```



- Stenst prófið?

Smíðum Order-
Information hlut

Fyrsta einingaprófið

- Skrifum einfaldasta mögulega próf

```
class TestOrderInformation(unittest.TestCase):  
    def test_create_order_information(self):  
        # Test the creation of an object  
        order_info = OrderInformation()  
  
if __name__ == '__main__':  
    unittest.main(exit=False)
```

- Nei! Enginn kóði sem útfærir OrderInformation



Rule #1: Your test should always
FAIL before you implement any code.

Brugðist við falli á prófinu

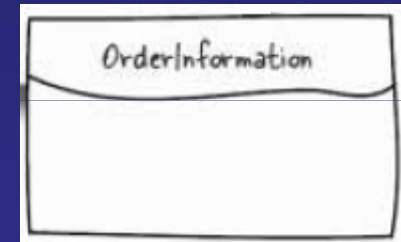
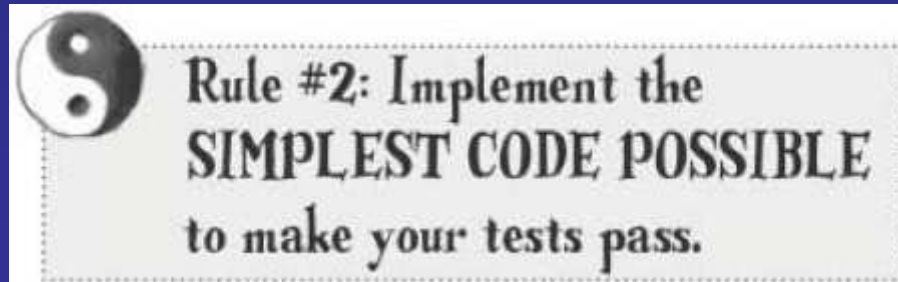
- Skrifum **einfaldasta** mögulega kóða sem stenst prófið

```
class OrderInformation(object):  
    pass
```



Gerir ekkert!

Geymum í
orderinfo.py



UML klasarit

OrderInformation klasinn

- Klasinn gerir ekkert sem stendur
- Skrifum einingapróf fyrir klasann

```
import unittest
import orderinfo

class TestOrderInformation(unittest.TestCase):
    def test_create_order_information(self):
        order_info = orderinfo.OrderInformation()
    def test_order_information(self):
        # Test get/set methods
        order_info = orderinfo.OrderInformation()
        order_info.set_cust_name("Steinn")
        order_info.set_drink_desc("2x Espresso")
        self.assertEqual(order_info.get_cust_name(), "Steinn")
        self.assertEqual(order_info.get_drink_desc(), "2x Espresso")
if __name__ == '__main__':
    unittest.main(exit=False)
```

← Próf eiga að vera sérhæfð

OrderInformation klasinn

```
class OrderInformation(object):  
    def set_name(self, name):  
        self._name = name  
    def set_drink_desc(self, drink_desc):  
        self._drink_desc = drink_desc  
    def get_name(self):  
        return self._name  
    def get_drink_desc(self):  
        return self._drink_desc
```

- Erum við að gleyma `__init__`?

↑
Hér væri ekki úr vegi
að nota `property()`

Helstu kostir TDD

- Eykur tiltrú á kerfinu
 - Próf dekka nánast allan kóða
- Hjálpar við skipta verkinu upp í viðráðanlegar einingar (verkpætti)
 - Hjálpar líka við að skýra notendasögur
- Minni tími fer í aflúsun á seinni stigum verkefnis
- Vitum hvenær verkinu lýkur

Helstu gallar TDD

- Notagildi er takmarkað þegar **kerfispróf** frekar en einingapróf gegna lykilhlutverki
 - Notendaskil
 - Gagnagrunnar ☹
- Geta gefið falskt öryggi
- Talsverður tími getur farið í viðhald á prófum

Ítarefni

- Grein um TDD á Wikipediu
 - http://en.wikipedia.org/wiki/Test-driven_development
- Einingapróf á gagnagrunnum (sett með til gamans en óþarft fyrir þetta verkefni)
 - <http://www.dbunit.org/intro.html>