Testidel põhinev arendus

Märt Kalmo

Materjalid

- http://enos.itcollege.ee/~mkalmo/tpa/

```
public class MySort {

     public static void mySort(List inputList) {
          // do some sorting stuff
     }
}
```

```
public class MySort {

	public static void main(String[] args) {
		List list = Arrays.asList(3, 6, 1, 4, 0);

		mySort(list);

		System.out.println(list);
	}

	public static void mySort(List inputList) {
		// do some sorting stuff
	}

}
	// [0, 1, 3, 4, 6]
```

```
public static void main(String[] args) {
      List list1 = Arrays.asList(3, 6, 1, 4, 0);
      List list2 = Arrays.asList(5, 4, 3, 2, 1);
      List list3 = Arrays.asList(-1, -3, -3, 2);

      mySort(list1);
      mySort(list2);
      mySort(list3);

      System.out.println(list1);
      System.out.println(list2);
      System.out.println(list3);
}
// [0, 1, 3, 4, 6]
// [1, 2, 3, 4, 5]
// [-3, -3, -1, 2]
```

```
public class MySort {
     public static void mySort(List inputList) {
          // do some sorting stuff
     }
}
```

```
public static void main(String[] args) {
    List list4 = Arrays.asList(-1, 0, 1, 0, -1);

    mySort(list4);

    System.out.println(list4);
}
```

```
public class MySortTest {
    @Test
    public void testMySort() {
        List list1 = asList(3, 6, 1, 4, 0);
        List list2 = asList(5, 4, 3, 2, 1);
        List list3 = asList(-1, -3, -3, 2);

        MySort.mySort(list1);
        assertThat(list1, is(asList(0, 1, 3, 4, 6)));

        MySort.mySort(list2);
        assertThat(list2, is(asList(1, 2, 3, 4, 5)));

        MySort.mySort(list3);
        assertThat(list3, is(asList(-3, -3, -1, 2)));
    }
}
```

```
// [0, 1, 3, 4, 6]
// [1, 2, 3, 4, 5]
// [-3, -3, -1, 2]
```

- Valmis töölaud
- Äralõhkumise kaitse

Kood: 60 min

Test: 5 min

Suhe: 8%

Mis on testitavus?

Sisendid

[3, 6, 1, 4, 0] - asList(3, 6, 1, 4, 0)

[5, 4, 3, 2, 1]

[-1, -3, -3, 2]

Vs. paljudest objektidest koosnev graaf

Väljundid

[0, 1, 3, 4, 6]

[1, 2, 3, 4, 5]

[-3, -3, -1, 2]

Vs. Html, Pdf, void meetodid

Sõltuvused

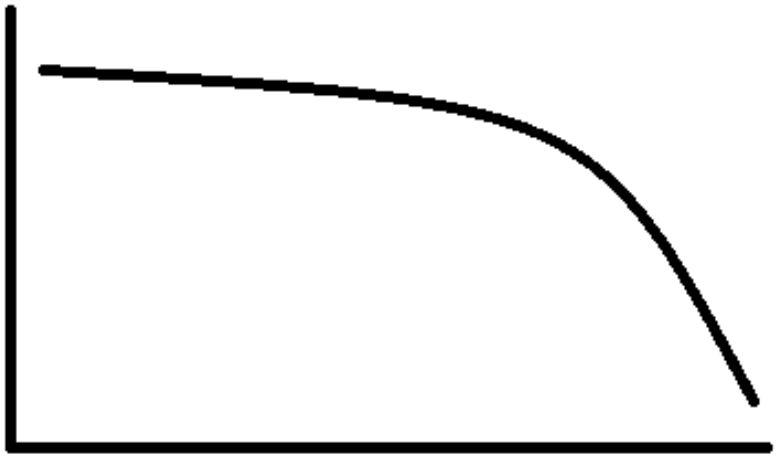
- Muu kood, teegid
- Andmebaas
- Veebiteenused
- Riistvara
- Operatsioonisüsteem
- jne.

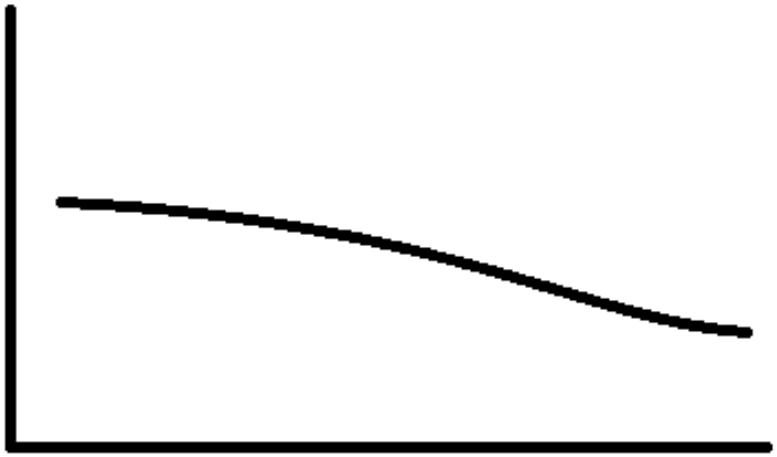
Oskused (disain)

- Selgelt defineeritud funktsionaalsusega meetodid
- Sõltumatud vs. sõltuvad meetodid

Milleks?

- 8% -> 100% ja rohkem
- Nt. Sqlite omab 1000x rohkem testkoodi





Vertikaal: funktsionaalsuse lisandumine
Horisontaal: aeg



Festivalist / Alafestivalid / Žüriid / Auhinnad / Festivali reeglid / Akrediteerimine / Festivalide koostöö / Kontakt

Pimedate Ööde Filmifestival
Gonsiori 21
10147 Tallinn

Tel.: 631 46 40
Faks: 631 46 44
E-mail: poff@poff.ee
info@poff.ee

Uudised

25.10.2006 | Tallinna Kinomajas peetakse Rahvusvahelist Animapäeva »

24.10.2006 | Ungari filmipäevad lõpetab M. Jancsó legendaarne film »

22.10.2006 | Tuntud tundmatud ungari filmid »

PÖFFi postiljon

Tunne ennast mugavalt - liitu **PÖFFi meililistiga** ja meie e-postiljon toob värsked festivali-uudised otse sinu postkasti!
» Liitu

POLIISI KIIRMÜÜK - PÕHIEESEME RISKIANDMED

Toode | Sõiduk | **Põhieeseme riskiandmed** | Kindlustusvõtja | Kontaktandmed

SÕIDUKI ANDMED

Uuenda | Kustuta | Vali

Registrimärk:	314MKU	**Reg.tunnistuse nr:**	ED068028
Mark:	MAZDA	**Ehitusaasta:**	2007
Mudel:		**Võimsus:**	108
Kaubanduslik nimetus:	6	**Registrimass:**	1940
VIN-kood:	JMZGY19F671459640	**Istekohtade arv:**	5
Kategooria:	Sõiduauto (M1)	**Eestis arvel oleku koht:**	

RISKIANDMED

Kaitse klass:	Tavasõiduk	**Ohtlikveos:**	Ei
Piirkond:	- Vali -	**Kliendi registrikood:**	38104014919
Sõiduki kategooria:	Kõik kategooriad	**Kliendi liik:**	Eraisik
Sõiduki kasutusala:	23 - Sõiduautod (rendiautod)	**Residentsus:**	Eesti
Mootori võimsus:	108	**Kliendi vanus:**	30
Registrimass:	1 940	**Kliendi riskikoefitsient:**	0,62
Istekohtade arv:	5	**Suvepoliis:**	Ei
Takso:	Ei		

RISKIANDMED (LKF)

Kaitse number:	
Kinnitamise tunnus:	
Kaitse kehtivuse algus:	

KINDLUSTATUD RISKID JA OMAVASTUTUS

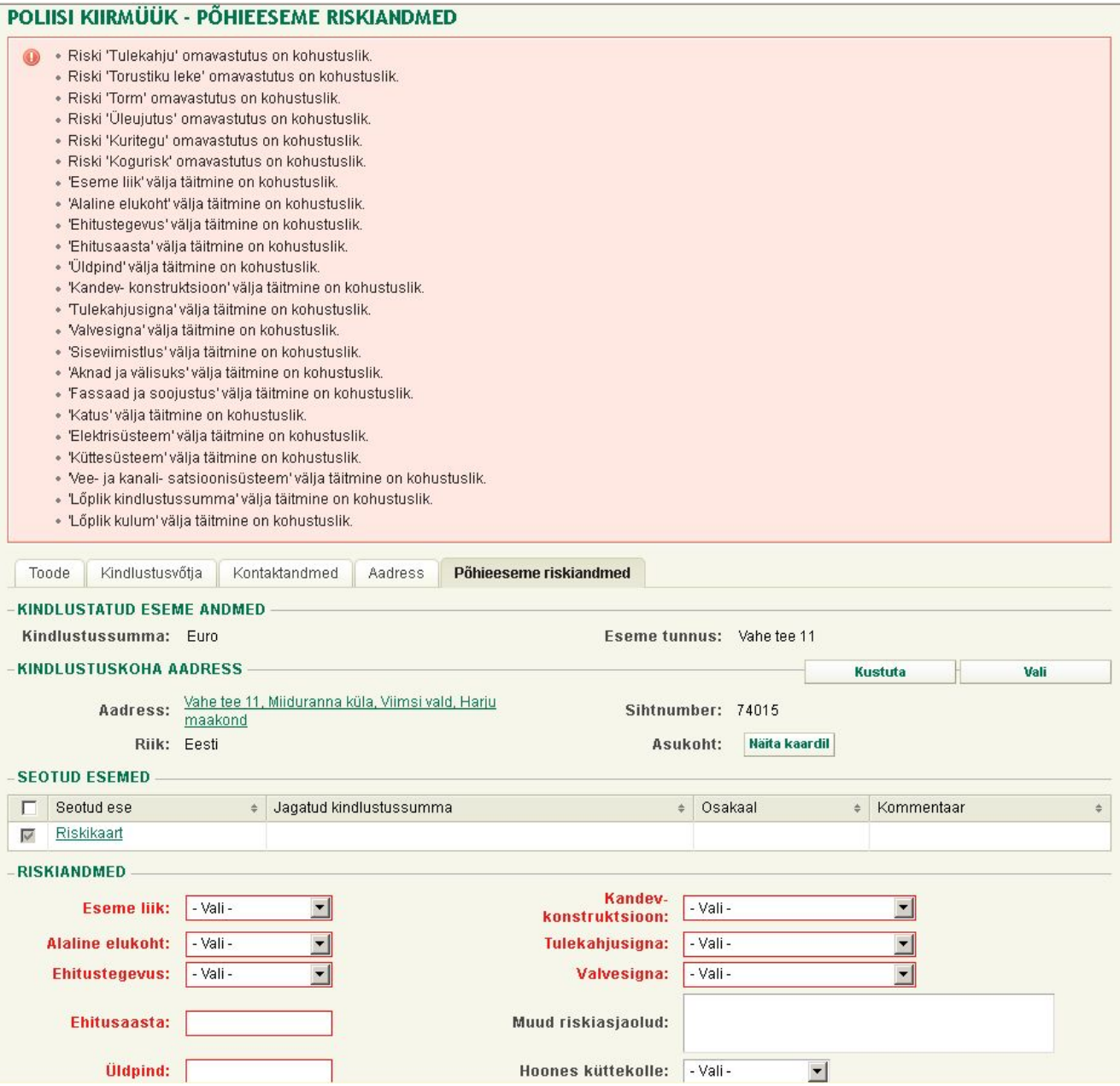
Tagasi | Edasi

Demo - string kalkulaator

- “” -> 0
- “1” -> 1
- “1, 2” -> 3
- null -> IllegalArgumentException

Kasu automaattestidest

1. aeg



POLIISI KIIRMÜÜK - PÕHIEESEME RISKIANDMED

- Riski 'Tulekahju' omavastutus on kohustuslik.
- Riski 'Torustiku leke' omavastutus on kohustuslik.
- Riski 'Torm' omavastutus on kohustuslik.
- Riski 'Üleujutus' omavastutus on kohustuslik.
- Riski 'Kuritegu' omavastutus on kohustuslik.
- Riski 'Kogurisk' omavastutus on kohustuslik.
- 'Eseme liik' välja täitmine on kohustuslik.
- 'Alaline elukoht' välja täitmine on kohustuslik.
- 'Ehitustegevus' välja täitmine on kohustuslik.
- 'Ehitusaasta' välja täitmine on kohustuslik.
- 'Üldpind' välja täitmine on kohustuslik.
- 'Kandev- konstruktsioon' välja täitmine on kohustuslik.
- 'Tulekahjusigna' välja täitmine on kohustuslik.
- 'Valvesigna' välja täitmine on kohustuslik.
- 'Siseviimistlus' välja täitmine on kohustuslik.
- 'Aknad ja välisuks' välja täitmine on kohustuslik.
- 'Fassaad ja soojustus' välja täitmine on kohustuslik.
- 'Katus' välja täitmine on kohustuslik.
- 'Elektrisüsteem' välja täitmine on kohustuslik.
- 'Küttesüsteem' välja täitmine on kohustuslik.
- 'Vee- ja kanali- satsioonisüsteem' välja täitmine on kohustuslik.
- 'Lõplik kindlustussumma' välja täitmine on kohustuslik.
- 'Lõplik kulum' välja täitmine on kohustuslik.

Toode | Kindlustusvõtja | Kontaktandmed | Aadress | Põhieeseme riskiandmed

KINDLUSTATUD ESEME ANDMED

Kindlustussumma: Euro

Eseme tunnus: Vahe tee 11

KINDLUSTUSKOHA AADRESS

Kustuta | Vali

Aadress: Vahe tee 11, Miiduranna küla, Viimsi vald, Harju maakond

Sihtnumber: 74015

Riik: Eesti

Asukoht: Näita kaardil

SEOTUD ESEMED

Seotud ese	Jagatud kindlustussumma	Osakaal	Kommentaar
Riskikaart			

RISKIANDMED

Eseme liik: - Vali -

Alaline elukoht: - Vali -

Ehitustegevus: - Vali -

Ehitusaasta:

Üldpind:

Kandev-konstruktsioon: - Vali -

Tulekahjusigna: - Vali -

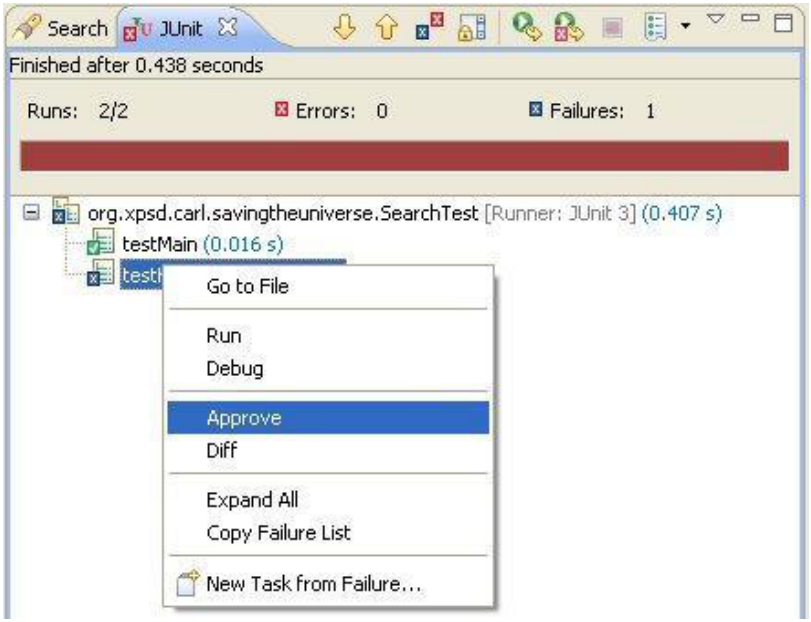
Valvesigna: - Vali -

Muud riskiasjaolud:

Hoones küttekolle: - Vali -

1. aeg

- Tavaliselt testitakse klikates
- Mõelda tuleb ikka – trükkimine pole pudelikaelaks
- Võimalik turueelis
- Tulemusi ei pea tingimata pliiatsi ja paberi abil valmis arvutama



1. aeg

- Arvestada tuleb kogu aega
- Google-i hinnangul:
  - $5 to fix a bug immediately
  - $50 after full build
  - $500 after integration test
  - $5000 after system test

2. lihtsustab muutmist

- Annab kindluse, et me olemasolevat funktsionaalsust katki ei tee
- Ei pea nii põhjalikult süvenema
- Ka väikesed muudatused vajavad testimist
- Ka väikesed veaparandused vajavad testimist
- Live süsteemid
- Arenduskeskkonna puudumine

3. parem disain

- Kasutatavamad liidesed
- Loetavamad, arusaadavamad, taaskasutatavamad meetodid/klassid

4. parem kvaliteet

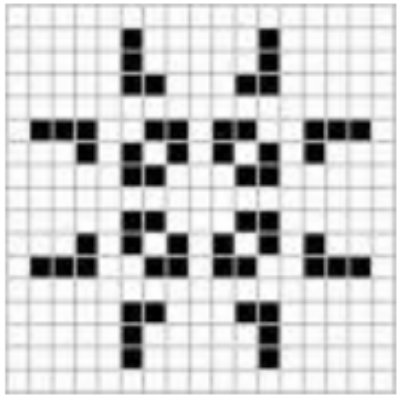
- Kvaliteet on meeldiv kõrvalefekt
- Kvaliteeti eeldatakse, sellega ei kaubelda
- Aeg-ajalt nõutakse vastupidist

Ülesanne – string kalkulaator

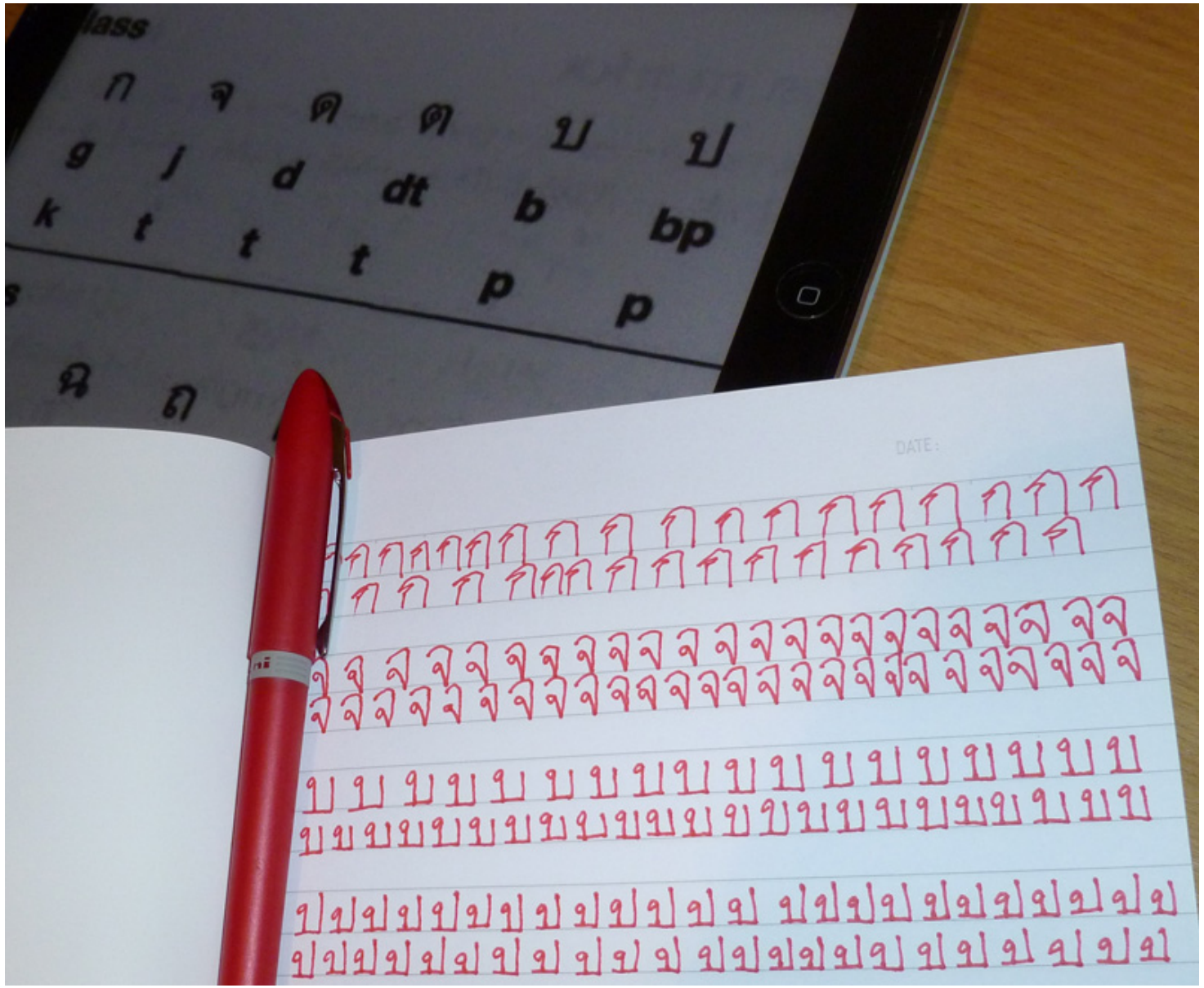
- “” -> 0
- “1” -> 1
- “1, 2” -> 3
- null -> IllegalArgumentException

Harjutamine

- 10 aastat programmeerimise kogemust ei tähenda iseenesest midagi
- Enesevaatlus (introspection), koodi lugemine, code kata
- Lihtsate asjade puhul on kergem head disaini teha
- Code retreat. kata 6x45min. iga kord sama probleem. Game of life



Harjutamine



Test Driven Development

Kent Beck



Test Driven Development

- On pigem disaini kui testimise metoodika
- Kirjutamise käigus õpib juurde
- Kui ei tea, mis on hea disain aga kirjutad testid enne (Lihtne setup, selged ootused jne.), siis tuleb hea disain iseenesest.

Example Driven Development, Test Driven Design, BDD, Executable Example

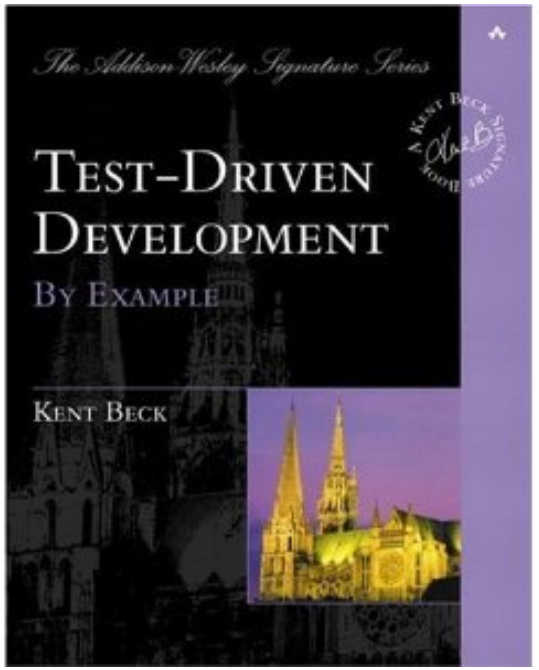
TDD näide

30 eek

15 eek

1 eur

sum: 4 eur



```
Object[][] moneyArray = new Object[][] {
            {30, "eek"},
            {15, "eek"},
            {1, "eur"}
};

int sum = 0;
for (int i = 0; i < moneyArray.length; i++) {
    if (moneyArray[i][1].equals("eek")) {
          sum += (Integer) moneyArray[i][0] / 15;
    } else if (moneyArray[i][1].equals("eur")) {
          sum += (Integer) moneyArray[i][0];
    }
}

System.out.println(sum + " eur");
```

TDD Töötsükkel

- lisa test (ühtki rida koodi ei tohi kirjutada, ilma et selleks test oleks)
- käivita testid ja jälgi, et uus test ebaõnnestub (testi testimine)
- lisa (minimaalne) kood või muuda seda
- käivita testid ja jälgi, et kõik testid õnnestuvad (enne igat käivitamist arva kas õnnestub)
- Refaktoreeri

TDD Töötsükkel

- Copy, understand, improvise

```
@Test
public void addingMoney() {
        Money money = new Money(2);
        money.plus(3);
        assertEquals(5, money.amount);
}
```

- klass Money
- konstruktor
- meetod plus
- väli amount

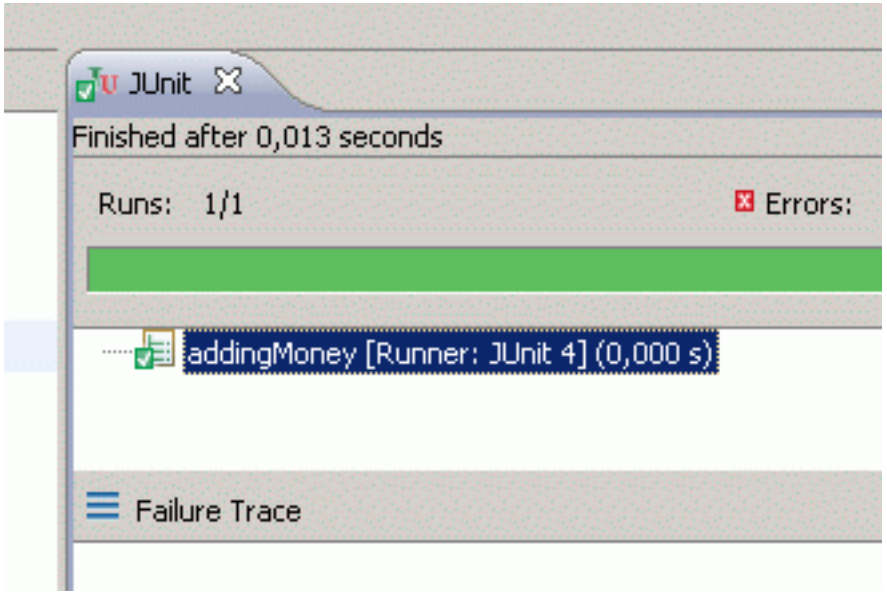
```
class Money {

        public int amount;

        public Money(int i) {}

        public void plus(int i) {}
}
```

```
@Test
public void addingMoney() {
	Money money = new Money(2);
	money.plus(3);
	assertEquals(5, money.amount);
}
```




```
public class Money {
	public int amount = 5;
	public Money(int i) {}
	public void plus(int i) {}
}
```

Üllatav TDD-st

- Testid katavad vähe funktsionaalsust
- Väiksed ja koledad muudatused
- Teste käivitatakse väga tihti

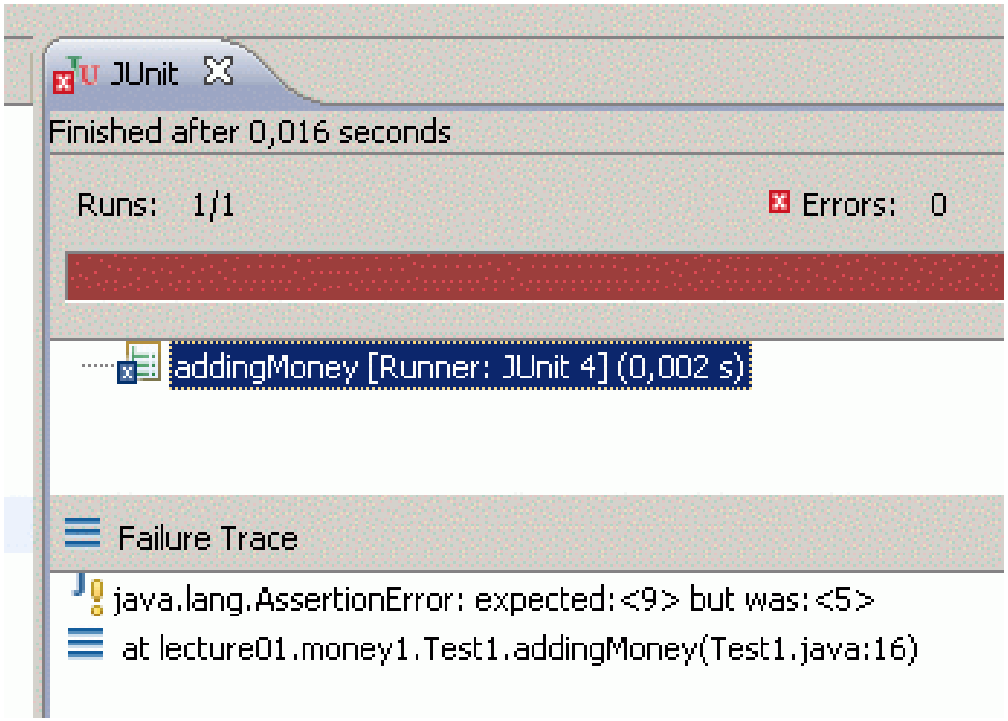
Üllatav TDD-st

Am I recommending that you actually work this way? No. I’m recommending that you

be able to work this way.

— Kent Beck

```java
@Test
public void addingMoney() {
        Money money = new Money(2);
        Money money2 = new Money(4);
        money.plus(3);
        money2.plus(5);
        assertEquals(5, money.amount);
        assertEquals(9, money2.amount);
}
```



```
public class Money {
	public int amount;

	public Money(int amount) {
		this.amount = amount;
	}

	public void plus(int amount) {
		this.amount += amount;
	}
}
```

```
Money money = new Money(2);
money.plus(3);

doSomething(money);

System.out.println(money.amount);
```

```
@Test
public void addingMoney() {
      Money money = new Money(2);
      Money sum = money.plus(3);
      assertEquals(5, sum.amount);
}
```

```
public class Money {
      …
      public Money plus(int amount) {
            this.amount += amount;
            return null;
      }
}
```

```
public class Money {
        …
        public Money plus(int amount) {
                return new Money(this.amount + amount);
        }
}
```

```
@Test
public void addingMoney() {
  Money money = new Money(2);
  Money sum = money.plus(3);
  assertEquals(5, sum.amount);
}
```

```
@Test
public void addingMoney() {
   assertEquals(new Money(5), new Money(2).plus(3));
}
```

```
@Test
public void testEquals() {
  assertTrue(new Money(5).equals(new Money(5)));
  assertFalse(new Money(0).equals(new Money(1)));
}
```

```java
@Override
public boolean equals(Object obj) {
  if (!(obj instanceof Money)) return false;

  Money money = (Money) obj;
  return amount == money.amount;
}
```

```java
@Test
public void addingMoney() {
  Money money = new Money(2);
  money.plus(3);
  assertEquals(5, money.amount); // money.getAmount()
}

@Test
public void addingMoney() {
  assertEquals(new Money(5), new Money(2).plus(3));
}
```

```
public int amount; -> private int amount;
```

```java
@Test
public void addDifferentCurrencies() throws Exception {
  new Money(4, "EUR").plus(3, "EEK");
}
```

```java
@Test
public void addDifferentCurrencies() throws Exception {
  new Money(4, "EUR").plus(3, "EEK", "EUR");
}
```

```java
@Test
public void bankConvertsCurrencies() throws Exception {
  Bank bank = new Bank();
  Money eur = new Money(1, "EUR");
  assertEquals(new Money(15, "EEK"), bank.convert(eur, "EEK"));
}
```

```
@Test(expected = IllegalArgumentException.class)
public void wrongCurrencyThrowsException() {
    new Money(0, "EUR").plus(new Money(1, "EEK"));
}
```

```
public class Money {
  private int amount;
  private String currency;

  public Money(int amount, String currency) {
    this.amount = amount;
    this.currency = currency;
  }

  public Money plus(Money money) {
    if (!currency.equals(money.currency)) {
      throw new IllegalArgumentException(
              "can't add different currencies");
    }
    return new Money(amount + money.amount, currency);
  }

  @Override
  public boolean equals(Object obj) {
    if (!(obj instanceof Money)) return false;
    Money money = (Money) obj;
    return amount == money.amount
            && currency.equals(money.currency);
  }
```

```
public class Bank {

  public Money convert(Money money, String targetCurrency) {

    if (money.getCurrency().equals(targetCurrency))
      return money;

    return new Money(money.getAmount() / 15, "EUR");
  }
}
```

```
@Test
public void sumMoney() {
  Bank bank = new Bank();
  Money sum = new Money(0, "EUR");

  for (Money each : Arrays.asList(
          new Money(30, "EEK"),
          new Money(1, "EUR"),
          new Money(15, "EEK"))) {

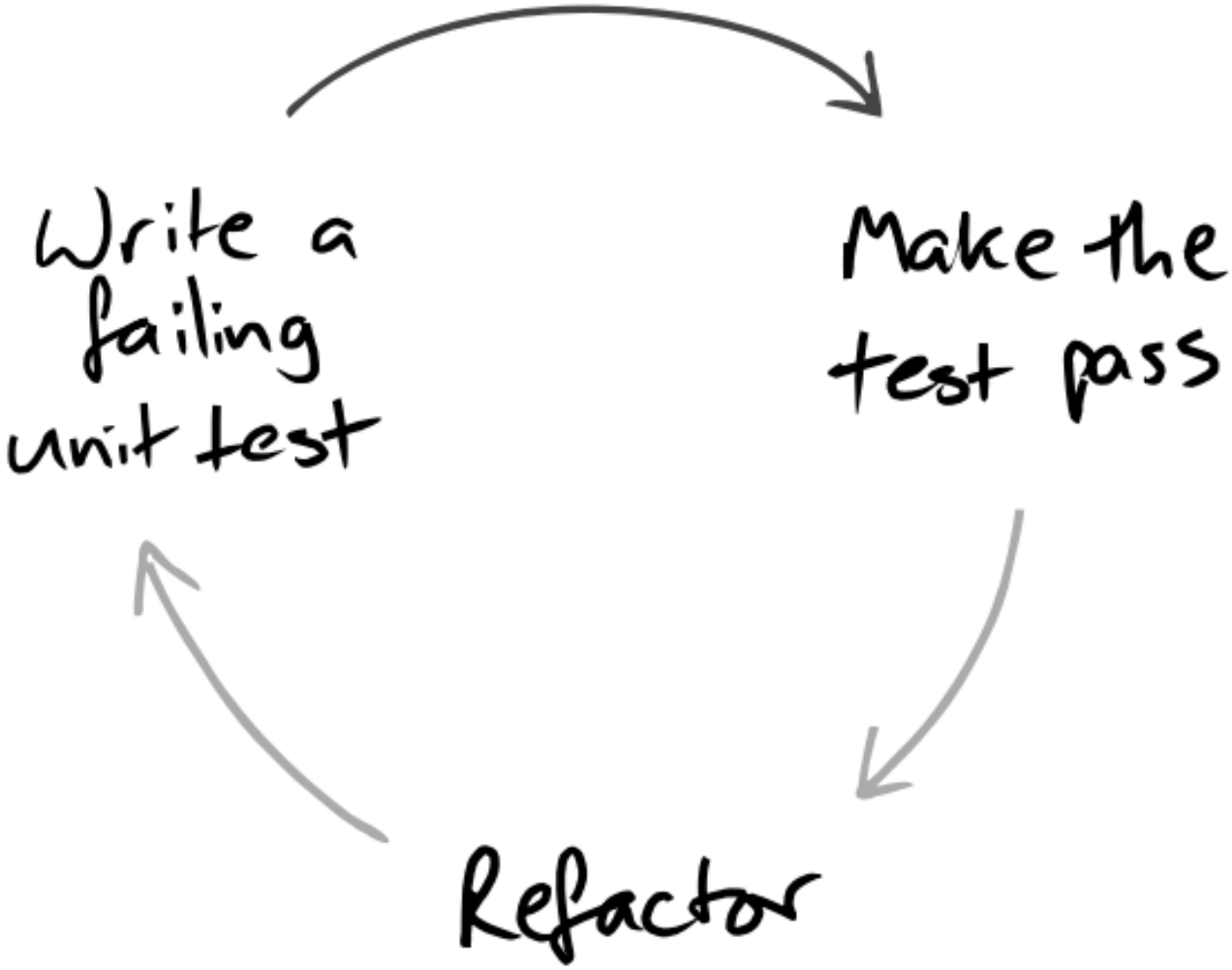
    sum = sum.plus(bank.convert(each, "EUR"));
  }

  assertThat(sum, is(new Money(4, "EUR")));
}
```

Ülesanne – pinu (stack)

- LIFO
- push, pop, peek, size
- Andmestruktuuriks on massiiv, mille suurus antakse konstruktoris ette.

TDD Töötsükkel



```
public Stack(Integer capacity) {
    st = new Integer[capacity];
    pointer = -1;
}

public Integer size() {
    return pointer + 1;
}

public void push(Integer i) {
    st[pointer + 1] = i;
    pointer += 1;
}

public Integer pop() {
    if (pointer >= 0) {
        pointer -= 1;
        return st[pointer + 1];
    } else {
        throw new IllegalStateException();
    }
}
```

Nimetamine

- test1()
- pushPop()
- pushOneAndTwoAssertSizeIsTwo()

Nimetamine

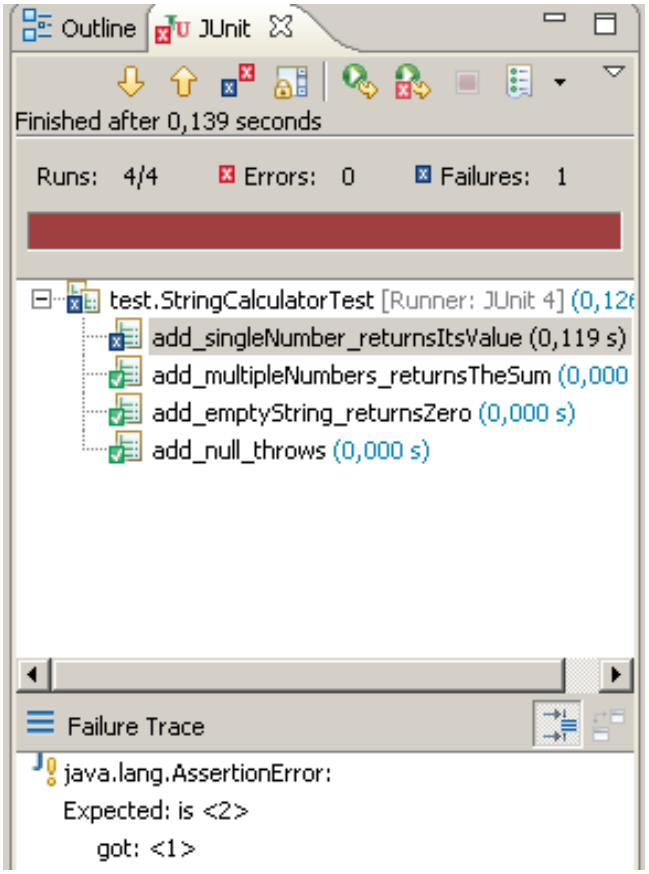
- pushIncreasesStackSizeByOne()
- removingAllElementsLeavesStackEmpty()
- poppedStackElementsAreSameAsPushed()

Nimetamine

What to call your test is easy—it's a sentence describing the next behaviour in which you are interested.

– Dan North

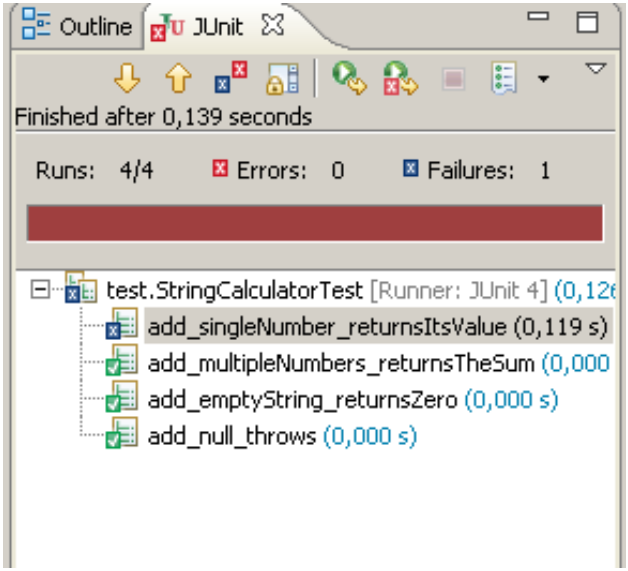
Nimetamine


Outline
JUnit
Finished after 0,139 seconds
Runs: 4/4
Errors: 0
Failures: 1
test.StringCalculatorTest [Runner: JUnit 4] (0,12
add_singleNumber_returnsItsValue (0,119 s)
add_multipleNumbers_returnsTheSum (0,000
add_emptyString_returnsZero (0,000 s)
add_null_throws (0,000 s)
Failure Trace
java.lang.AssertionError:
Expected: is <2>
got: <1>

Kasu automaattestidest vol 2

5. toimib dokumentatsioonina

- Käititatav dokumentatsioon
- Specification By Example
- Spec-ist otsimine on aeganõudev
- Mis on taotluslik ja mis juhuslik
- Legacy code – kood ilma testideta



6. lihtsustab integreerimist

- Mida rohkem on koodi mida me ei usalda seda aeganõudvam on vigade otsimine

7. vähendab riski

- Algsete arendajate lahkumine
- Kui pikaks kujuneb vigade parandamise faas?
- Kas edaspidi suudame vajalikku funktsionaalsust lisada?

8. programmeerija rahulolu

- Kindlus
- Sujuvus, stressi maandamine
- Vähem vearaporteid
- Ületundide vältimine

Automaattestimine praktikas

- Pole kuigi laialt levinud
- Teatakse ja peetakse vajalikuks

Miks ei kasutata?

- Alati ei olegi kasulik
- Teadmatus, oskamatus
- Töötamise tavasid on raske/riskantne muuta
- Infot reaalsete projektide kohta pole

Kes kasutavad?

- Codeborne, Hansapank, Justiitsministeerium, Nortal (vähesel määral)
- Üksikuid teste leiab enamustes projektides

100% pole eesmärk

- Programmeerimine peab lihtsamaks ja mugavamaks muutuma, mitte vastupidi

Testid peaksid eelkõige olema kohtadele:

- mis tihti muutuvad
- mis on keerulised
- mida on aeganõudev käsitsi testida

Millele teste kirjutada (näited)

- Hinna arvutamine (muutuv)
- Maksegraafiku genereerimine/muutmine (keeruline)
- Valideerimine (aeganõudev/tüütu käsitsi testida)

Paarisprogrammeerimine



Paarisprogrammeerimine

- Tuleb XP-st
- Variatsioon – ping-pong programmeerimne

Ülesanne - Rpn kalkulaator

- Rpn – Reverse Polish Notation
- 1 + 2 → 1 2 +
- (1 + 2) * 3 → 1 2 + 3 *




```
RpnCalculator c = new RpnCalculator();
c.setAccumulator(1);
c.enter();
c.setAccumulator(2);
c.plus();
assertThat(c.getAccumulator(), is(3));
```

Testkoodi disaini põhimõtted

- Hea disain on oluline (haldamine). Sellega tuleb pidevalt tegeleda.
- Nimed on pikad
- Duplikatsiooni on pisut rohkem. Põhirõhk loetavusel
- Testides pole hargnemisi

Koodi kaetus testidega

- Kvantitatiivne mõõde
  (nt. testide arv pole piisavalt hea)
- Koodi kaetust hindavad tööriistad

Lausekaetus
(statement coverage)

```
public static Object[] reverse(Object[] a) {

      if (a == null)
            return null;

      Object[] b = new Object[a.length];
      for (int i = 0; i < a.length; i++) {
            b[i] = a[a.length - 1 - i];
      }
      return b;
}
```

Otsusekaetus (decision coverage)

```
private void meetod1(int a) {

  int c = 0;

  if (a == 1) {
    c++;
  }

  return a / c;
}
```

Kaetus

- Lausekaetus on hea, Otsusekaetus parem jne.
- Ükski kaetus ei taga, et testid leiavad kõik vead! (no asserts, puuduv kood)
- Kaetuse näitaja puudus: illusioon korrektsusest

Ülevaade edasisest

- Puhas kood ja refaktoreerimine
- Sõltuvuste haldamine
- Testimise mustrid ja võtted