TDD Test Driven Developement

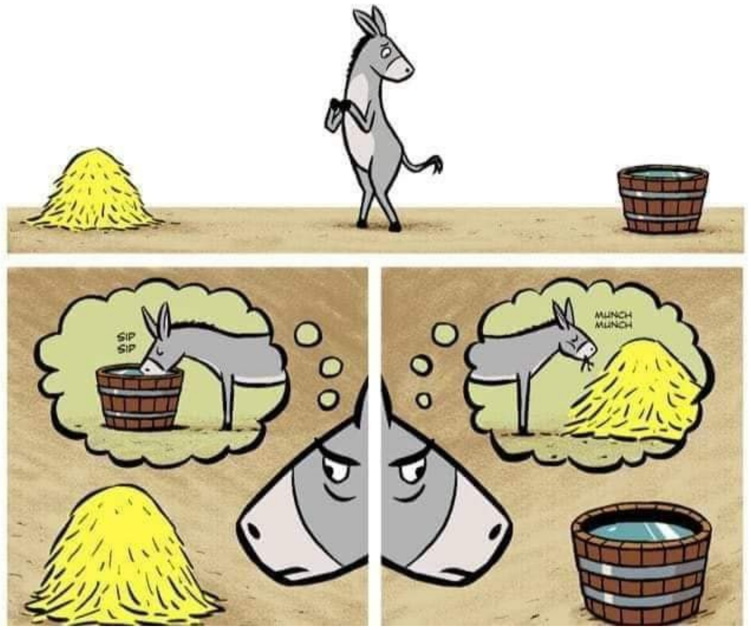
Measure 7 times, cut once

Plan

- Introduce test first and code first concept
- For a given task illustrate code first and test first implementations
- Detalize TDD values

What is first: code or test?

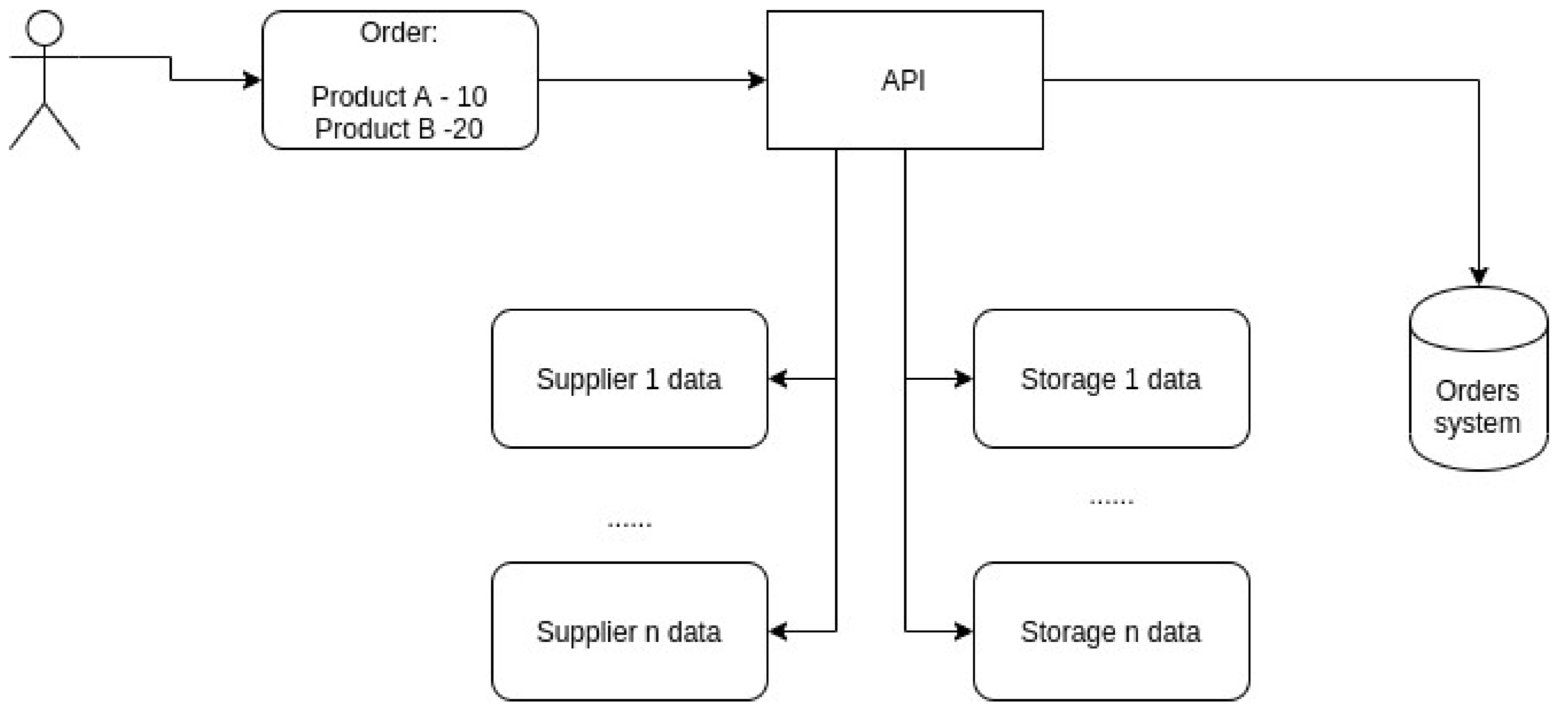
- Code first – complex test containing a lot of mocks.
- Test first – simple test may contain no mocks



Order registering example

- Must register order
- During registration must **reserve** products from storages and mark what is needed to get from suppliers
- Available products amounts are **distributed** between **storages**, marked with identifiers and by **projects** also marked with identifiers
- Each supplier is marked with a virtual **storage** and **project** id
- Result order lines must be identified by correct **storage** and **project** identifiers and **reservation** amount values.

Order registering illustration



Implementation without test

- Because implementation is done without test, whole task is implemented in a complete code, which includes
  - Storages and projects configurations load
  - Order identification fix
  - Order validation and availability check
  - Prices loading and setting
  - Correspondence schemes
  - Delivery data
  - Order date and order validity date set
  - Resulting order registering to database
  - Response making and returning.

Why I needed a test?

- User told (lied) that order registration is **incorrect**
- To proof that user right or wrong I needed to reproduce the situation, which requires a lot of work and would have a **short term** value
- Or I could just create unit test for my code, which would have a **long term** value.

Resulting test looks like

- 235 lines of code
- 20 assert lines
- About 20 mock statements
- To be able to assert the resulting order, should intrude a writting process using mocking techniques.

Values of a test-first approach

- Compliance to the business requirements
- Enhancement of the application structure
- Enchancement of a programmer work methodology
- Team management
- System stability

Testing scenarios

- 3 articles : one at local storage, one at supplier, third – missing.
- 1 article , three units: one at local storage, one at supplier, third – missing
- 1 article, many units – available at different local storages and projects

3 articles : each in a separate state

Order:

Sku	Amount
P1	1
P2	1
P3	1

Amounts in storages

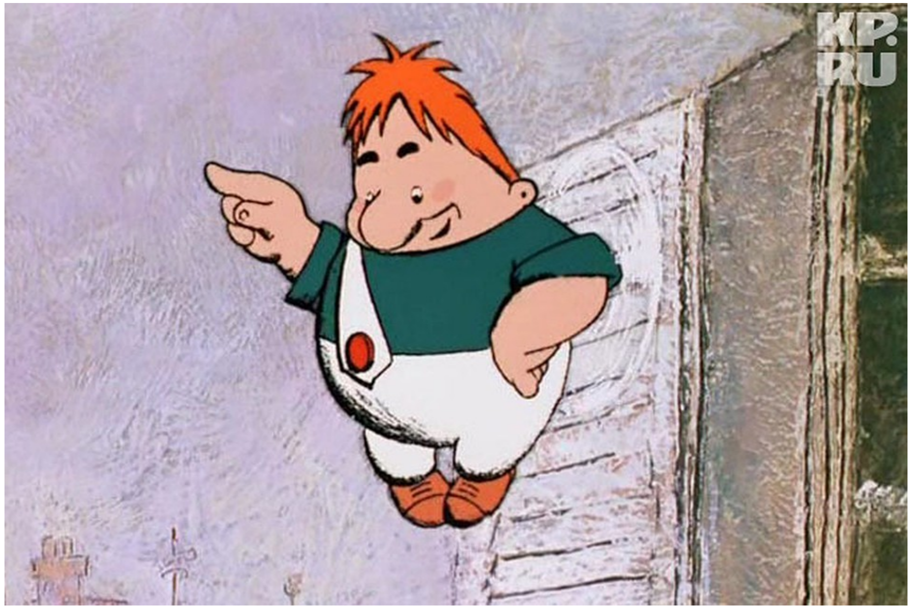
sku	Storage	Project	amount
P1	101	1001	1

Amounts at provider

sku	Provider	Amount	Related storage	Related project
P2	T1	1	9901	99901

Expected result order

Sku	amount	Reserved amount	Storage	Project
P1	1	1	101	1001
P2	1	0	9901	99901
P3	1	0	Null	null



One article three units: each in a separate state

Order

Sku	Amount
P1	3

Amounts in storage

sku	storage	Project	amount
P1	101	1001	1

Amounts at supplier

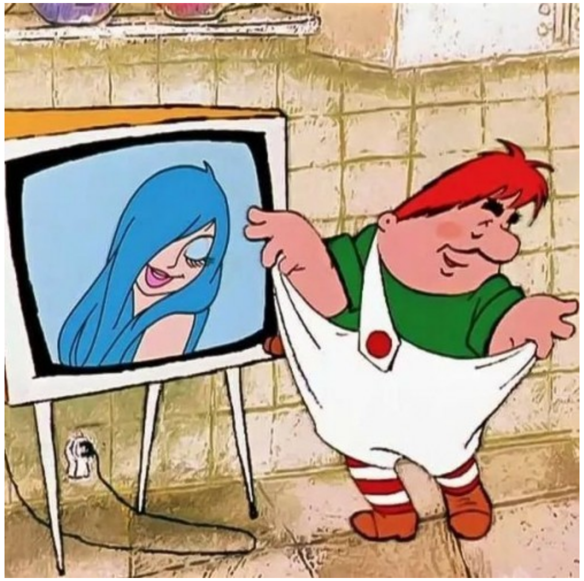
sku	Supplier code	Amoutn	Related storage	Related project
P1	T1	1	9901	99901

Expected order

Sku	amount	Reserved amount	Storage	Project
P1	1	1	101	1001
P1	1	0	9901	99901
P1	1	0	Null	null

Test code

```php
public function testCalculateOrder(
    array $localAmounts,
    array $supplierAmounts,
    OrderReservationInfo $order,
    OrderReservationInfo $expectedOrder
) {
    $rezOrder =
OrderHandlerPartial::calculateOrderByAvailableAmounts(
        $localAmounts,
        $supplierAmounts,
        $order
    );
    $this->assertEquals($expectedOrder, $rezOrder);
}
```



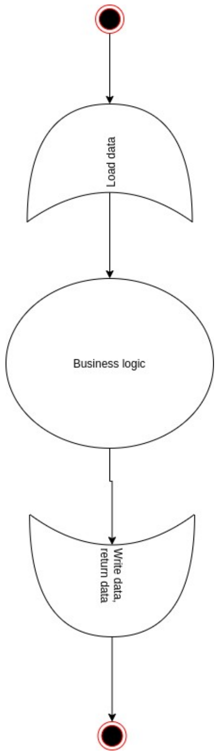
Data function

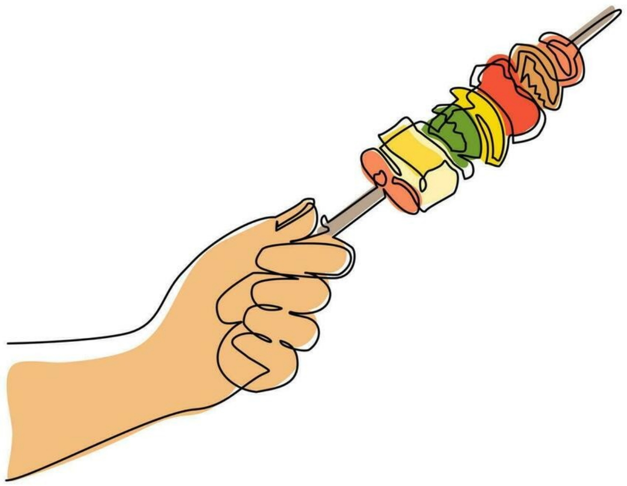
```
'testG' => [
   'localAmounts' => [
      (new  ProductAmountInStorage())
         ->setNomNr('P1')
         ->setLiko(1)
         ->setSandelisID(101)
         ->setProjektasID(1001)
   ],
   'supplierAmounts' => [
      new TiekejoPrekesLikutis('T1', 'P2', 1, 9901, 99901)
   ],
   'order' => (new OrderReservationInfo())
      ->addLine((new OrderReservationLineInfo())
            ->setProductNomNr('P1')
            ->setAmount(1))
      ->addLine((new OrderReservationLineInfo())
            ->setProductNomNr('P2')
            ->setAmount(1))
      ->addLine((new OrderReservationLineInfo())
            ->setProductNomNr('P3')
            ->setAmount(1)),
   'expectedOrder' => (new OrderReservationInfo())
      ->addLine((new OrderReservationLineInfo())
         ->setProductNomNr('P1')
         ->setAmount(1)
         ->setReservedAmount(1)
         ->setSandelioId(101)
         ->setProjektasId(1001)
      )
```

Enhancement of a code structure

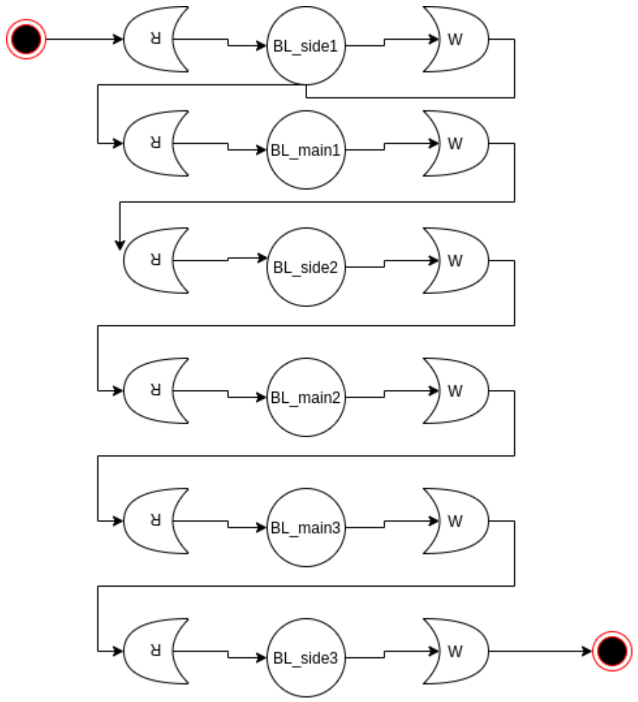
- DI correct enought structure
- Pure function only for testing
- Separate data loading and writing from business logic
- Identify the main functionality and separate from the additional functionality.

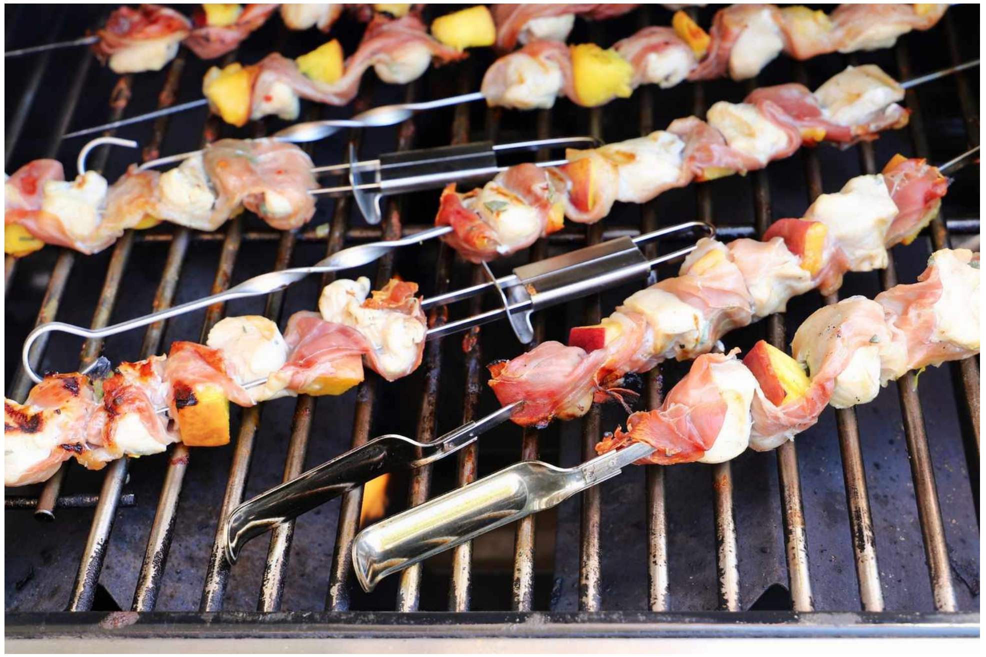
Execution flow simple



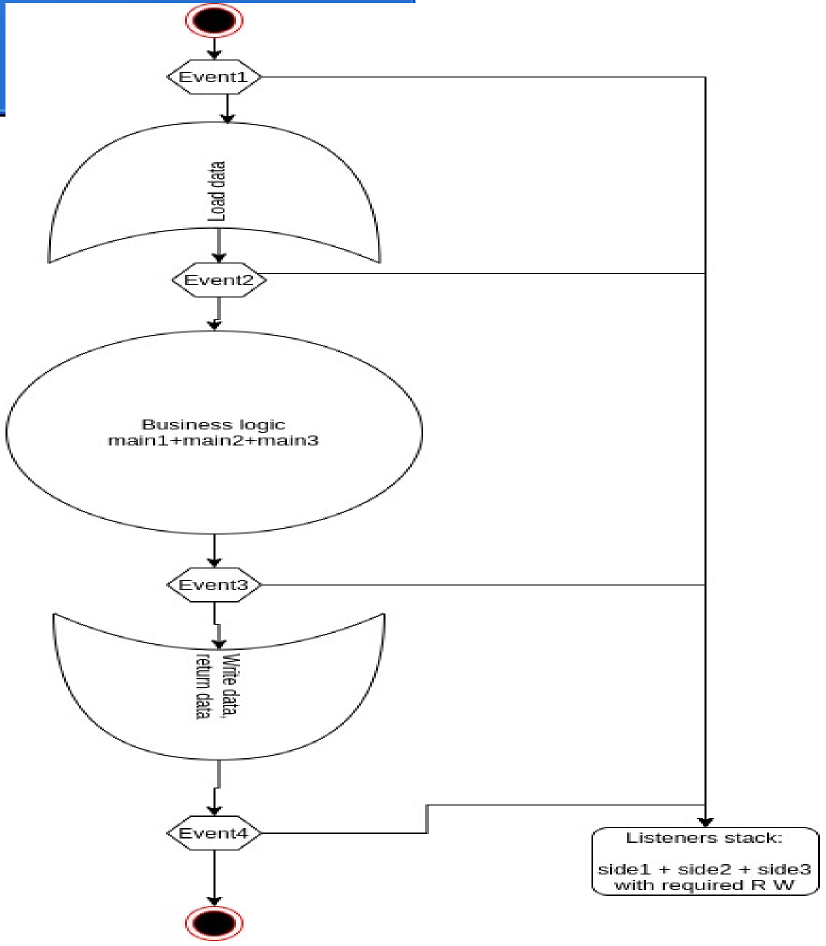


Real world execution flow

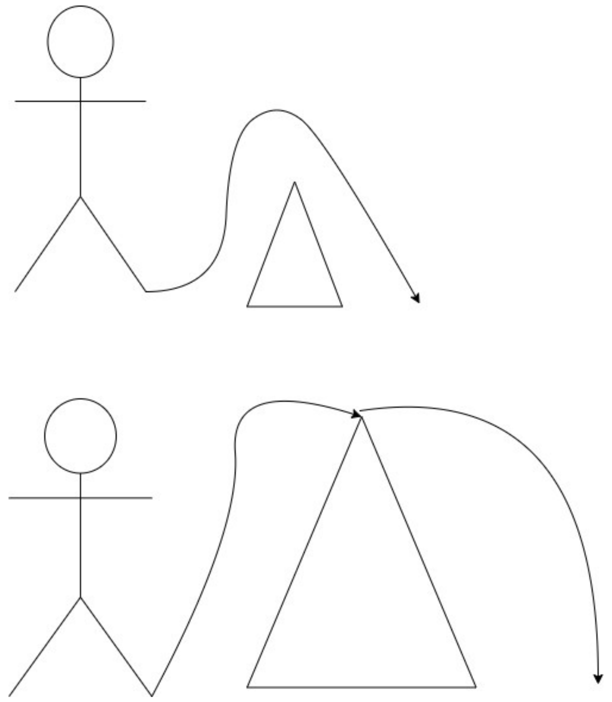




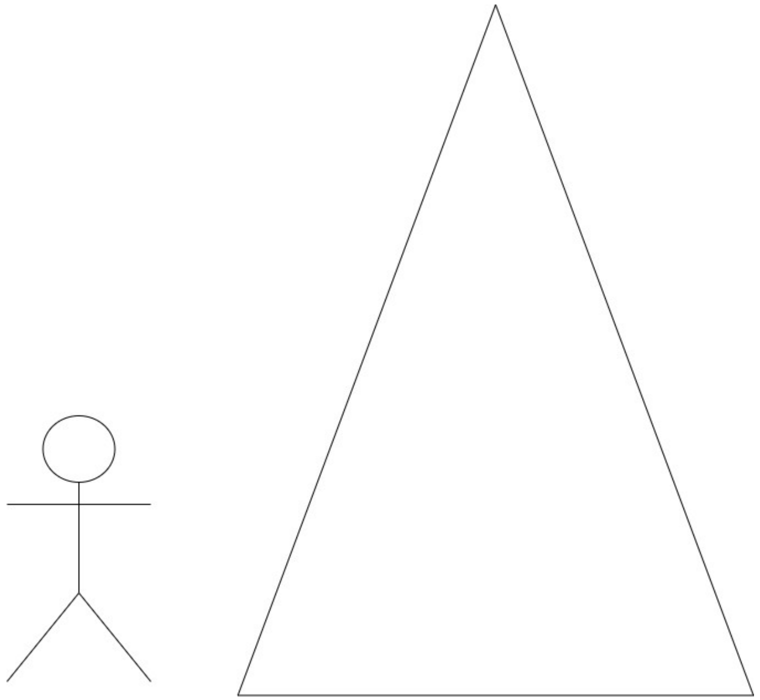
Planned execution flow.



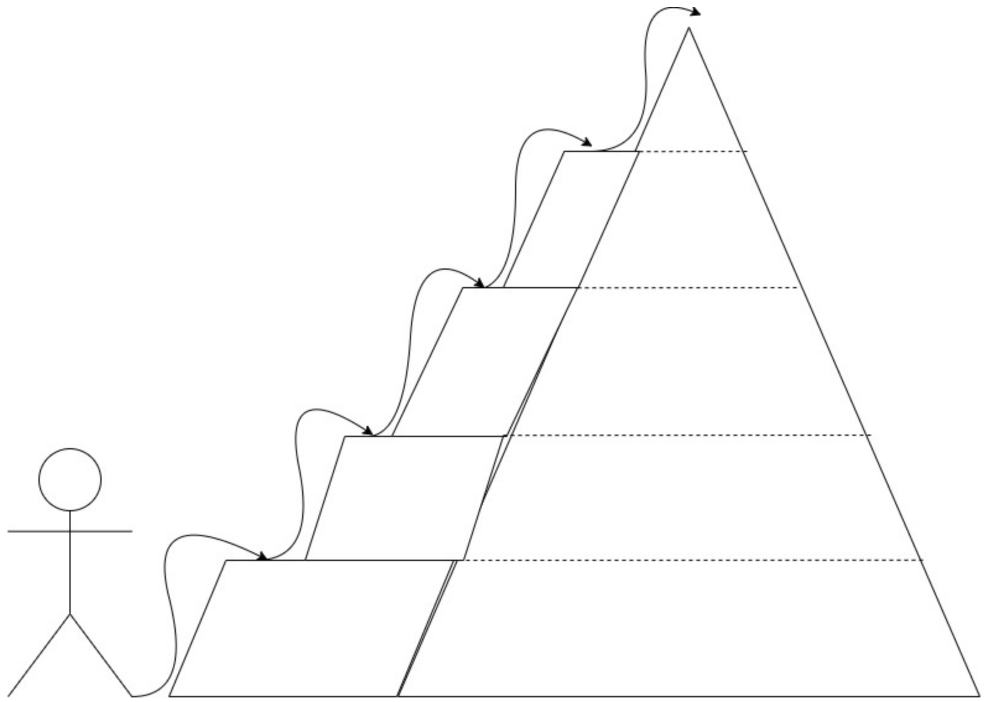
Enhancement of a programmer work methodology: easy task



Complex task



Complex task solution



Team management

- No need to make a precise code review when you have tests.
- Pure functions covered with test probably doesn’t contain nasty bugs. So making a function ‘pure’ restricts programmer to make a hidden code wich will contain bugs in future.
- Code depth is decreased, so it becomes easier to read.

System stability

- Makes possible to refactor system without breaking functionality.
- Automatic tests in deployment prevents from releasing bugs in a live fystem.

Conclusions

- Test **before** code better than test **after** code, but test **after** code still **better** than **no** tests.
- Tests makes possible to check **compliance** with the business requirements
- Tests makes application **code structure** better by separating a **read/write** code from a calculation/business logic code; making **pure functions** and increasing the **code readablity**.
- Tests makes a programmer possible to **split** a task to smaller parts and make **commits more often**.
- Test permits to manage team by **avoiding** too precise **code reviews** and denying to put **hidden code** and **hidden bugs**.
- Tests makes system more **stable** both in **deployment** stage and in a **refactoring** tasks.