Prof. Dr.-Ing. Peter Liggesmeyer
M. Sc. Sebastian Müller

TU Kaiserslautern
Dept. of Computer Science
AG Software Engineering: Dependability

Software Quality Assurance (WS16/17)

Problem Set 4

Due: in exercise, 14.12.2016

Problem 1: Testing Object-oriented Software

Given is a simplified specification of ATM operation sequences:

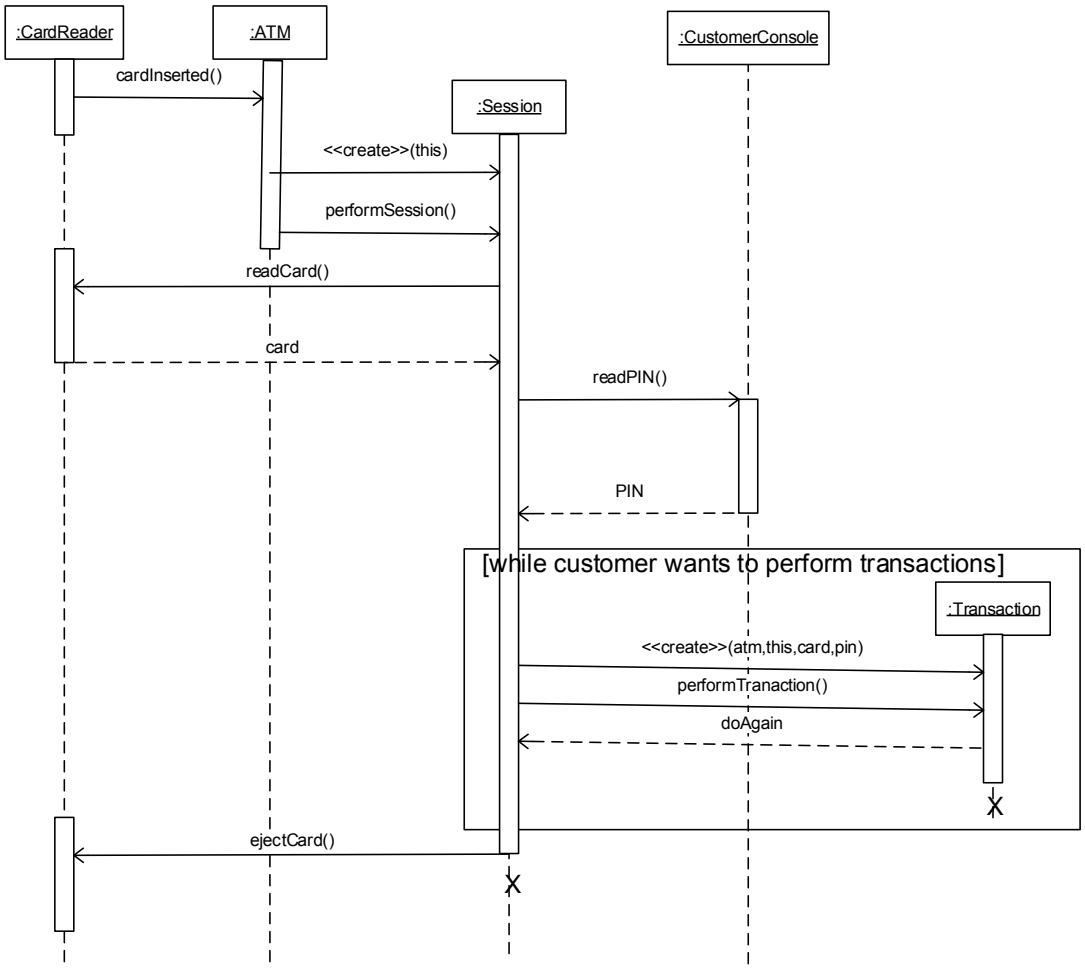
1. Customer enters card into the card reader
2. Customer console requesting pin entry is displayed
3. An interleaved sequence of digits is proved by the ATM
4. If user enters wrong pin three times, the card will be retained
5. Withdrawal transaction is created and performed
6. Cash dispenser dispenses approved amount to customers
7. Session ends if no transaction is active
8. Card is returned to customer
9. Possibility of cancellation by customer

The following additional requirements are to be considered:

1. PINs are four-digit number sequences, meaning ≥ 0001 and ≤ 9999
2. Withdrawal is only possible by choosing the menu options that are displayed by the customer console.
3. *For this cash machine, it is only possible to input digits.*

a) Please draw the **state transition diagram** for one session context according to this description. A session sequence diagram is shown below, which will additionally help you to derive the state chart.

Session Sequence Diagram

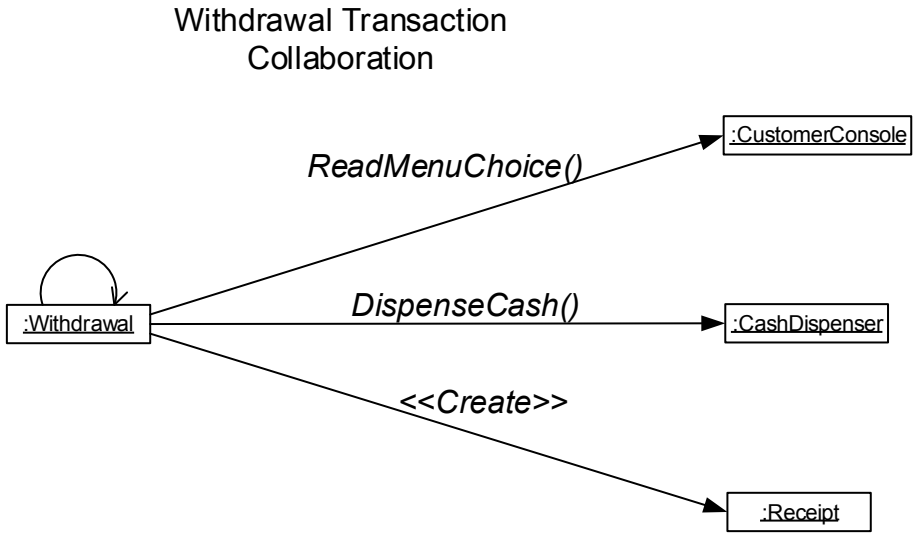


b) Class test (unit test)
After an initial analysis, the following components (individual classes) are identified:

1. Operator panel
2. Card reader
3. Customer console, consisting of a display and keyboard
4. Network connection to the bank
5. Cash dispenser
6. Receipt printer
7. Session
8. Transaction

Attached you will find an excerpt of a standard for Test Documentation. Please read it and derive a unit test plan for this ATM system that includes the classes mentioned

c) Integration Test
Test of the intersection of the classes *customerConsole* and *Withdrawal* (sub-class of Transaction) via the message *ReadMenuChoice()* (see attached diagram Withdrawal Transaction Collaboration).



Interface specification of the operation *ReadMenuChoice()*

- The operation *ReadMenuChoice()* expects a non-negative value ($10 \leq \text{value} \leq 500$). It specifies the value of the bill in euros.
- The operation has the return value of the amount selected by the customer.

Interface parameter of the calling routine at the *CustomerConsole* device

- The following values can be chosen by the customer: *ReadMenuChoice()*: 20, 50, 100, 150, 200.

1. Please determine the assertion of this interface parameter for the integration test and derive the equivalence classes and test cases for input and output interfaces.
2. Please determine the equivalence classes and test cases for the operation *readPIN()* as well (with Boundary Value Analysis, *readPIN()* returns 4-digit number put in by the customer).
3. Attached you will find a template (Integration Test Plan), which is generated from a real project. Please read through it and derive an integration test plan for this ATM system, with an emphasis on the interfaces *ReadMenuChoice()* and *readPIN()*.

d) System Test

1. For a functional system test, what functions need to be tested? How do you ensure the completeness of your test set?
2. How do you test system performance (reaction time and the like)?

Now the customer of this ATM system demands to have an additional menu option for the user, the "300" option. This option is obviously not included in the available menu options, and should be added to both the *CustomerConsole* and the *Withdrawal* classes. The interface specification is altered, and *ReadMenuChoice()* is overwritten accordingly as well.

3. Do you need new test cases for this new menu option? Why? If so, which test cases do you need? Should the assertion be changed?

Problem 2: Function-oriented Test

- The system to be tested consists of two components: `Thrower` and `ThingsToBeThrown`. Thrower calls the method `throw()` from `ThingsToBeThrown`.
- The original documentation of the component `ThingsToBeThrown` is as follows (it is a few years old already):
- "First shalt thou take out the Holy Pin, then shalt thou count to three, no more, no less. Three shall be the number thou shalt count, and the number of the counting shall be three. Four shalt thou not count, neither count thou two, excepting that thou then proceed to three. Five is right out. Once the number three, being the third number, be reached, then lobbest thou thy Holy Hand Grenade of Antioch towards thy foe, who being naughty in My sight, shall snuff it."[1]

a) First you have to test the interface between the classes `Thrower` and `ThingsToBeThrown` via the message `throw()`.

Interface specification of the operation `throw()`:

- the operation `throw()` expects a `delay` in seconds and a flag `safetyRemoved`, the delay is an integer value between 0 and 10, the flag is a Boolean value

- it has no output

Interface parameter of the calling routine

- The following values can be chosen by the thrower: the flag `safetyRemoved` can be true or false, as delay 2, 3, 4, 5 can be chosen

Determine the assertion of the parameter delay for the integration test and derive the equivalence classes for the interface.

b) Determine test cases for the operation `throw` based on a boundary value analysis of the input parameters.

c) For a functional system test, what functions need to be tested? How do you ensure the completeness of your test set?

d) The system requirements have been changed. Now the thrower component can choose a delay of 2, 3, 4, 5, 8. Which new test cases do you need? Which ones can be reused? How does the assertion change?

[1] You should know where this citation stems from, otherwise you can look it up on Wikipedia.