IN2013, Week 9 – Test Cases (use case based testing and class testing with jUnit)

Model Answers

Dr Peter T. Popov

22nd November 2018

In this tutorial we will practice with use case based testing and with jUnit testing.

Question 1.

Consider the case-study BAPERS and the use case model of the system included in Appendix 1.

Develop test cases for all flows of the use case PlaceOrder. Pay attention to the fact that you cannot test extension use cases on their own – they only extend other use cases and therefore must be tested together with the use – case that they extend. This will affect:

- the system state before the test cases (i.e. setup including any possible extensions),
- the data to be used in the test cases (and their extensions) and
- how the state of the system changes at the end of the test cases (these may be changes visible during the interaction with the system or may require some non-trivial checks, e.g. of data storage, etc.)

Model Answer:

Use case ID: TM02	**Use case name:** PlaceOrder
Test number: 1	
Objective: Test the ***main flow*** of the use case.	
Set up (must make sure that all conditions listed below is satisfied before the test): 1. BAPERS is on, a Receptionist is logged on. 2. An existing customer Peter Popov, ptp@csr.city.ac.uk, 02070408963 brings in a new job at 1 pm which must be completed urgently. 3. The job which will consist of the following tasks: 1, 4 and 6. 4. This customer does not have outstanding payments. 5. The last job ID used by the system is 100567. 6. The number of customers registered with BAPERS stands at 356.	
Expected results: 1. The customer record will be found. 2. The system will allow the Receptionist to enter the 3 tasks specified above. 3. The system will compute that job price is £107.30. 4. Completion time set by the system will be 7 pm (6 hours is the turnaround time for urgent orders.) 5. The job ID of the newly created job will be 100568 (printed on the label and recorded in the DB). 6. The number of customers registered with BAPERS will remain 356. 7. Once the job is placed it will be viewed as urgent and with correct details.	

Test:

1. The searches for Peter Popov and ptp@csr.city.ac.uk.
2. The system will show find a single record (the email is unique).
3. The Receptionist will enter tasks 1,4 and 6 in a loop of 3 iterations.
4. The Receptionist will specify that the job is urgent.
5. The system will compute a price of £107.30.
6. The Receptionist will confirm the price.
7. The system will print a label and a collection note with JobID = 100568.

Test record: The Receptionist can view the jobs under processing and find among them job 100568 with the details as specified.
The number of customers won't change as a result of taking in the new job (356 will remain unchanged).

Date: 24 September 2012	**Tester:** Peter Popov

Result: Fail. The jobID assigned was 100567. The rest was OK.

Test record: The expected outcome was observed.

Date: 25 September2012	**Tester:** Peter Popov

Result: Passed.

Other failures can also be observed, e.g. incorrectly computed price, completion time, etc.

Use case ID: TM02	**Use case name:** PlaceOrder

Test number: 2

Objective: Test the ***alternative flow NoCustomerAccountProvided*** of the use case.

Set up (must make sure that all listed below is satisfied before the test):

1. BAPERS is on, a Receptionist is logged on.
2. A new customer Peter Popov, ptp@csr.city.ac.uk, 02070408963 brings in a new job at 1 pm which must be completed urgently.
3. The job which will consist of the following tasks: 1,4 and 6.
4. This customer does not have outstanding payments.
5. The last job ID used by the system is 100567.
6. The number of customers registered with BAPERS – 356.

Expected results:

1. The customer record is ***not*** found.
2. A new customer record is created using the data provided above.
3. The system will allow the Receptionist to enter the 3 tasks.
4. The system will compute that job price is £107.30.
5. Completion time set by the system will be 7 pm (6 hours is the turnaround time for urgent orders.)
6. The job ID of the newly created job will be 100568 (printed on the label and recorded in the DB).
7. Once the job is placed it will be viewed as urgent and with correct details.
8. The number of customers registered with the system will be incremented and will be set to 357.

Test:

1. The Receptionist searches for Peter Popov and ptp@csr.city.ac.uk.
2. The system shows that no customer has been found.

3. The receptionist enters in a form the new customer data provided in the setup. 4. The system will create successfully a new customer record. 5. The Receptionist will enter tasks 1,4 and 6 in a loop of 3 iterations. 6. The Receptionist will specify that the job is urgent. 7. The system will compute a price of £107.30. 8. The Receptionist will confirm the price. 9. The system will print a label and a collection note with JobID = 100568.	
Test record: The expected behavior observed.	
Date: 24 September 2012	**Tester:** Peter Popov
Result: Passed.	

The most important part of the test cases is the data, which ***must be specific****. Provided the system is working correctly the data provided in the setup and in the test must activate the flows being tested: the main or the alternative.*

In the specific example make sure that you understand why only one test case is used to test both extensions. We have two extensions SearchCustomer and CreateCustomerAccount and typically we would use two separate test cases to test each of the extensions separately. We could have used the same setup but different Tests (i.e. the interaction of the actor with the system will involve one extension at a time).

An interesting test case would have been executing CreateCustomerAccount when the customer record already exists. The expected behavior in this case would be for the system to ***raise an exception****. Clearly, this test case would be an example of detecting deficiencies of the use case specification (as Extreme Programming proclaims), which in turn should lead to adding an important alternative flow to the specification.*

As an exercise at home develop this test case in detail and revise the use case specification to accommodate the need for the system to check and raise exceptions in case an actor tries to duplicate an existing customer record.

> *Provided a DB is designed with properly specified keys avoiding duplicates can be achieved using the build-in capabilities of a good databases server, i.e. no checks are necessary in the application software. Application software, however, still has to handle the exceptions raised by a server.*

Question 2.

Use the Java code generated by Visual Paradigm from BAPPERS design class diagram (in Tutorial 7) and create a Netbeans project from the generated code.

Using Netbeans built-in support for jUnit create test cases for the Task class and test the methods of the class following the steps listed below:

- Write the initialization code (@BeforeAll part of a jUnit test).
- Write test cases for start() and complete() methods of the Task class.
- Write Java code for these two methods. The code is fairly simple – should change the status of a :Task object.
- Run the tests, fix any bugs that you may encounter and repeat the tests until

the tests are passed.

Model Answer:

No model answer is provided here. Watch the video on Moodle for a worked out example of testing Task with jUnit.

Question 3.

Now consider the association of the Task class with the TaskDescription class and test this part of Task functionality. There are two options that you may consider:

- Extend the constructor so that it looks up for a :TaskDescription object which should be linked with a newly instantiated :Task object and assign the reference to this :TaskDescription object to an attribute refTD of Task which captures the association with :TaskDescription.
- You introduce a new setRefTD(td:TaskDescription), which will look up for a matching :TaskDescription and once a suitable one is found, will assign its reference to the attribute refTD.

Write test cases to test the chosen extension of Task and execute them until they are passed.

Discuss with a colleague the "oracles" that you used in all tests.

Model Answer:

Watch the video on Moodle for a worked out example of testing Task with jUnit.

Two java files are released on Moodle:

- *Task.java - minimalistic implementation of the class Task.*
- *TaskTest.java - the jUnit test for the Task.java class.*

The video shows how one can create a jUnit test in and how to run it in the Netbeans IDE.

Additional Assignments.

Extend the unit testing of BAPPER by writing jUnit tests for (some of) the methods of other classes, e.g. CustomerAccount and Job, write a Java code for these methods and apply the tests to them.

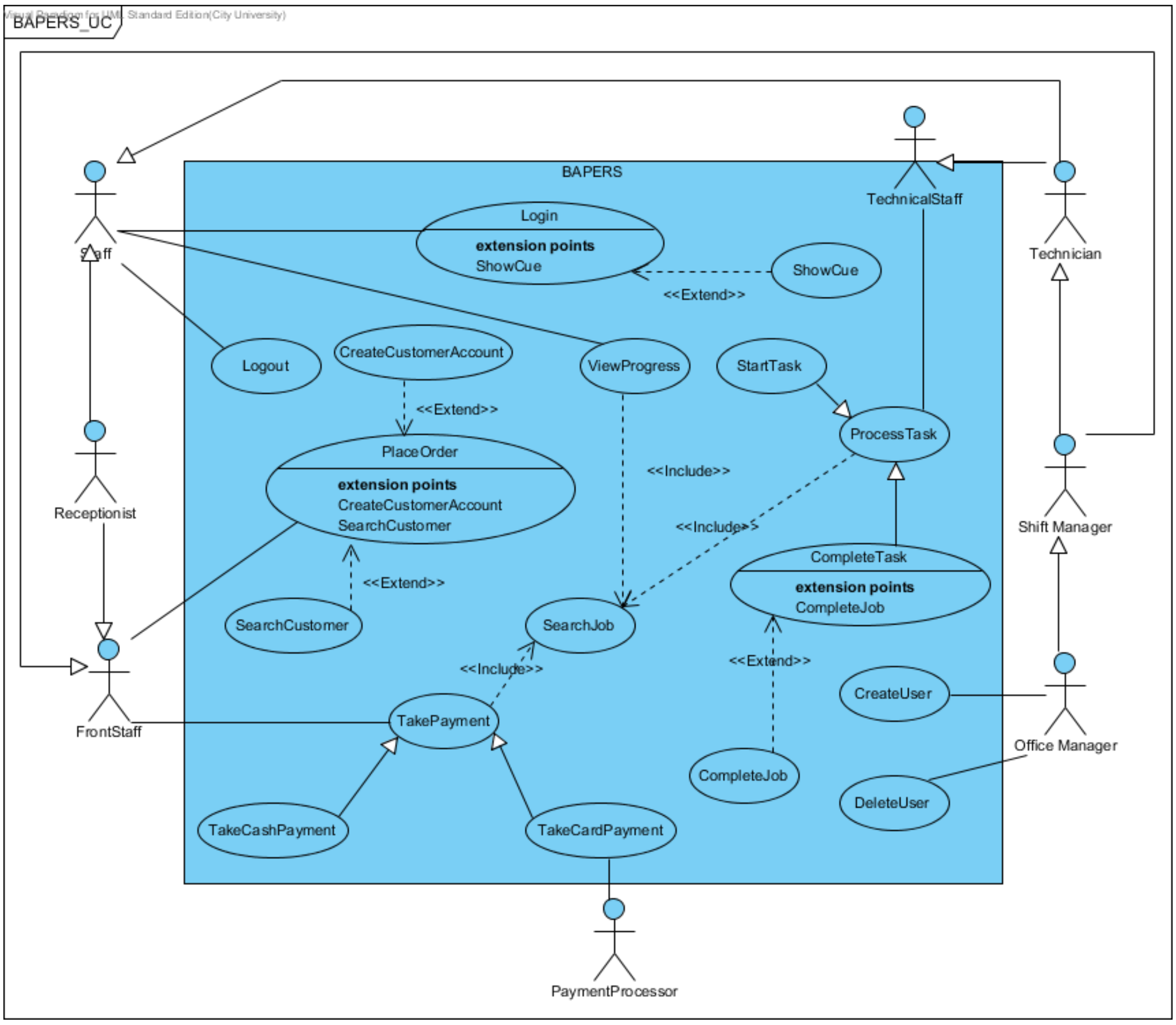
Model Answer:

No answer to this question is provided.

Created: 2nd November 2018

Last modified: 4th of November 2018

Appendix 1



ID: TM02	**Use case:** PlaceOrder
Brief description: The Customer brings in photographic material to be processed in "The Lab". A new Job sheet is made out with a unique ID, which lists the tasks to be performed, sets the completion time. The job is linked to a CustomerAccount. If necessary a new Customer Account is created.	
Primary actors: FrontStaff (Receptionist, Shift or Office Manager)	
Secondary actors: None	
Preconditions: 1. BAPERS is operational, the primary actor has logged in.	

Flow of events: 1. Actor activated the functionality for creating a new job Extension point: SearchCustomer Extension point: CreateCustomerAccount 2. Actor selects the chosen CustomerAccount. 3. The system assigns a unique ID to the Job and links the job to the chosen Customer account. 4. While there are more tasks to add to the job: 4.1. The actor selects a task from the list of available tasks and adds it to the job. 5. The actor selects the urgency of the job (either “urgent” or “normal”). 6. The system records the job completion time. 7. The system computes the price of the job. 8. The actor confirms that the job is completed. 9. The system prints a label (to be attached to the photographic material) with the job ID and a collection note[1] (with the job ID, the Customer name and the job completion time).
Postconditions: 1. The system has recorded a new job with its ID, Customer who placed it, urgency, completion time, price and list of tasks to be completed. 2. The system printed a label and a collection note.
Alternative flow: NoCustomerAccountProvided PrintingFailure

Alternative flow: NoCustomerAccountProvided
ID: TM02.1
Brief description: No CustomerAccount is provided.
Primary actors: FrontStaff
Secondary actors: None
Preconditions: The session does not contain a valid CustomerAccount.
Alternative flow: 2. The system prompts the actor that no valid CustomerAccount has been chosen. 3. The actor acknowledges the notification 4. The system removes the session.
Postconditions: None.

[1] This note is handed over to the customer as a proof that the photographic material has been collected for processing.

Alternative flow: PrintingFailure
ID: TM02.2
Brief description: Printer fails to print the required document: either is out of paper or some other problem occurs.
Primary actors: FrontStaff
Secondary actors: None
Preconditions: 1. The printer is out of paper or a technical fault has accured.
Alternative flow: 9. The alternative flow starts after step 9 of the normal flow 10. BAPERS displays a message: “Printer fault”. 11. If further details are known about the problem   The system prints further details.
Postconditions: None

Use case ID: TM03	**Use case:** CreateCustomerAccount
Primary actors: FrontStaff	
Secondary actors: None	
Segment 1 Precondition (extension point: CreateCustomerAccount**)** No account exists in BAPERS for the customer who brought in photographic material.	
Segment 1 flow: 1) Actor activates the functionality for creating new customer account. 2) The system places an empty customer record in the session. 3) The system opens a form for entering the customer details such as Name, phone number and email address. 4) The actor types in the required details and confirms their validity. 5) The system creates a new customer record in BAPERS. 6) If the system succeeds The system places the customer details in the session by adding a ‘current customer’ record in the session. Else Places an empty ‘current customer’ record in the session.	
Postcondition: 1. CustomerAccount details are stored in the ‘current customer’ record in the session.	

Use case ID: TM04	**Use case:** SearchCustomer

Primary actors: FrontStaff
Secondary actors: None
Segment 1 Precondition (extension point: SearchCustomer**)** Customer chooses the CustomerAccount search functionality by activated it.
Segment 1 flow: 1) The system offers a search form for customer to specify the search criteria (e.g. name, email, phone number or any combination of these) 2) While ***no more than one record*** is found 2.1 The actor specifies the search criteria (any combination of the three listed above will be acceptable) and confirms the validity of the search criteria. 2.2 The system undertakes a search based on the search criteria provided. 3) If a single customer record is found The customer record is placed in the ‘current customer’ record in the session. Else An empty record ‘current customer’ is placed in the session.
Postcondition: CustomerAccount details are stored in the ‘current customer’ record in the session.