**Build Application Components, Execute Unit Test and Update Source Code**

**Module Description**

In this module, the participants will:

* Code the missing logic for the application.
* Create stub code to isolate their implemented classes from other classes.
* Perform unit testing by running their JUnits against their classes (with the stub code in place) and checking the results. They will fix their code when needed (following a failed test).

**Scenario**

You will start your development environment and obtain a copy of the code that you will need to modify. Using the detailed design and the peer review checklist provided to you as a guide, you will:

* Write the logic in the Java class assigned to you by your team lead.
* Ensure that it compiles and executes.
* Review the code to determine if it conforms to the coding standards and the detailed design specifications.

The next step is the Peer Review. You (the author) will pass your completed work product to an available application developer (the reviewer) on your team. The reviewer will perform an un-facilitated Peer Review (i.e., they will review your deliverable on their own) of the work product, complete a Peer Review Feedback Form, and place the completed form in the document repository for you.

The work products that will be reviewed are:

* JUnit Test Scripts
* Stubbed code
* Any completed Java code classes/methods

Following the Peer Review, you, as the author of the reviewed work product, will review the Peer Review Feedback Form and make the required changes to the work product and document those changes by updating the Peer Review Feedback Form.

You will meet with the reviewer again if required to discuss which items were fixed (closed status) and which were not (rejected status). The rejected fixes will have a documented reason why the fix was rejected and the rejection must be approved and documented by the reviewer.

The updated work products and feedback forms are then checked in by the author. The Team will track the status and performance of the peer review process and use this to provide a status to the leadership.

As previously stated in the orientation, team status meetings are held periodically and this time is allocated for you to have a meeting with the rest of the team. The Team Lead will host this meeting and gather a status from you and other members of the team. The status is sent to Project Management on a regular basis and helps in communicating how well (or poorly) the project is progressing and can help the management determine if intervention is needed.

Once your JUnit Code and any reviewable source code has been peer reviewed, you will execute the JUnit scripts to run your code using the stubs.

You will then update your source code until all of the test cases work correctly. Make any changes to your test conditions, Stub Code, or JUnit Code as required.

Review your JUnit and Stub code using the peer review checklist to make sure it meets project standards.

Give your completed work product to another Application Developer (the reviewer) on your team so they can conduct an unfacilitated peer review (they check the deliverable on their own). You, in turn, will conduct a Peer Review for one of your colleagues, complete a Peer Review Feedback Form, and place it in the document repository. The work products that will be reviewed are:

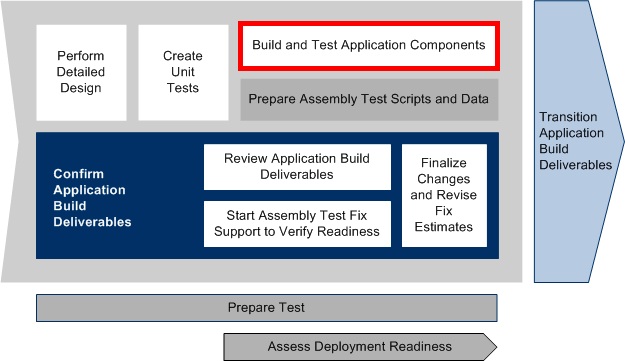
* JUnit Test Scripts
* Stubbed code
* Any completed Java code classes/methods

Upon completion of the review and any updates from the peer review, you will complete the Unit Test of your code. The Unit Test is complete when all test scripts have executed successfully. Once complete, you will check in all of the updated versions of your work products and code into the appropriate repositories.

As code by other developers is completed (coded, unit tested and peer reviewed), one or two team members will be designated to integrate the code. This requires replacing the stub code with code created by the team and successfully compiling it.

**ADM Activity Context Diagram**

Application > 4100 Build Application > 4188 Build and Test Application Components



**Roles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **ADM Task** | **ADM Responsibility** | **Description** |
| Project Manager | 4188 - Build and Test Application Components | Management Oversight | Provides guidance on the Team Status Report. |
| Team Lead | 4188 - Build and Test Application Components | Primary Performer | Designates Peer Review responsibilities.  Tracks the status and performance of the Peer Reviews.  Leads the Status meeting and compiles feedback.  Addresses and corrects any issues where possible. Escalates issues when appropriate. |
| Peer Review | Designates Peer Review Responsibilities.  Reviews work products as needed. |
| 4188 - Build and Test Application Components (Post Peer Review) | Assigns team members for code integration. |
| Developer | 4188 - Build and Test Application Components | Primary Performer | Implements the logic that is defined in the detailed design specification.  Checks that the logic meets the project's coding standards.  Peer Reviews other team members’ work products.  Makes, reviews, or rejects fixes, or performs updates to work products based on the results in the Peer Review Feedback Form.  Makes updates to the Peer Review Feedback Form.  Provides status on assigned tasks and activities including what has been completed, is in-progress or is delayed.  Communicates any issues that impact progress or productivity.  Executes Source code using automated test scripts.  Makes modifications to source code if actual results do not match expected results. |
| Peer Review | Peer Reviews other team members’ work products. |
| 4188 - Build and Test Application Components (Post Peer Review) | Integrate all completed code. |
| Team (All) | 4188 - Build and Test Application Components (Post Peer Review) | Primary Performer | Executes Source code using automated test scripts. |

**Participant Instructions**

**Build Application Components**

1. Review your assigned application design, AP333 Component/Class Design, focusing primarily on the pseudo code.

2. Within Eclipse, navigate to your copy of the FestivalPortal project (FestivalPortalParticipant), if you have not already done so.

3. The java classes and methods that need to be implemented are:

a. EventServiceImpl.java (SIMPLE)

ο checkEventsofVisitor()

ο getAllEvents()

ο updateEventNominations()

b. EventDAO.java(MEDIUM)

ο showAllEvents()

ο updateEventDeletions()

ο checkEventsofVisitor()

c. VisitorServiceImpl.java (SIMPLE)

ο createVisitor()

ο searchVisitor()

d. VisitorDAO.java (COMPLEX)

ο insertData()

ο searchUser()

ο registerVisitortoEvent()

ο registeredEvents()

4. Implement the logic for your assigned class methods as described in the detailed design pseudo code, keeping in mind:

a. You will need to complete the method body. Review the comments for more detail.

b. You must ensure the code compiles and appears to meet the standards in [TA243 Application Development Standards](https://accenture.desire2learn.com/content/enforced/9406-Pending/NCCC_Project_Work_Products/4_Tech_Arch/Release_1/AppDevStd_FERS_R1_REF.docx?_&d2lSessionVal=YnoL4daZGC8ne2RyGf0cWMrk8) and the application design specifications.

5. Follow the steps outlined in the Peer Review - Review Process section of the [Master Practice Primer](https://accenture.desire2learn.com/content/enforced/9406-Pending/NCCC_Project_Work_Products/1_PM/ADF_Java_2_0_Master_Practice_Primer_REF.docx?_&d2lSessionVal=YnoL4daZGC8ne2RyGf0cWMrk8) to review the source code using the Peer Review Checklist.

6. Note each method call in your code logic that calls another Java object:

a. For each method, add or update simple "stub" logic to that method that will generate the results that you wish to test.

b. Ensure you use the "stub" class and not the Java class that your teammates are implementing.

**Please Note:** You may refer to the section (Appendix C) on mock objects in the [Tools Primer](https://accenture.desire2learn.com/content/enforced/9406-Pending/Tools%20Primer/ADF_2_0_Tools_Primer.docx?_&d2lSessionVal=YnoL4daZGC8ne2RyGf0cWMrk8) for testing.

**Execute Unit Test**

7. In Eclipse, open the JUnit test script and the code that will be tested in the editor window.

8. Execute the JUnit scripts to run and test the code using the stub code (or skeleton code/classes that accept the input according to its own detailed design and give fake ‘correct’ output each time regardless of input).

9. If errors are given by JUnit, update the code to correct the problem.

10. Unit test is completed once all test cases are executed and the code provides the expected results in each case.

11. Document these percentages, as you will need them for the Build Completion Report.

12. If necessary (when new test conditions are discovered or existing test conditions are invalid), update the TCER document.

**Hints and Tips**

**Work Product Author:**

1. Don't be afraid to question the changes. Some feedback does get rejected when appropriate and when it makes sense.

2. Don't make, or indeed take, the feedback personally. It's about quality in the deliverables for the future.

3. Update the Status and Comments in the Peer Review Feedback Form provided by the Work Product Reviewer. This is proof of "due diligence" that the team has followed the project's quality standards/procedures by thoroughly fixing the work product or providing a reason not to.

**Work Product Reviewer:**

1. Fill out the Peer Review Feedback Form as an audit trail and proof of due diligence that the project's quality standards/procedures have been acknowledged and defects in the work product have been documented.

2. Only include feedback for worthwhile changes. While finding a lot of defects is good for the sake of quality, you need to ensure that you only document items that are truly defects.

3. Reflect on the concepts presented in the Peer Review and Productive Meetings Professional Skills training.