UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS 301
Mini Project Testing

Nardus van der Vyver u15012698
Minal Pramlall u13288157
Peter Rayner u14001757
Mandla Mhlongo u29630135

May 5, 2017

https://github.com/MandlaMashinini/Phase4_Testing

Contents

1 Service Contract Of Each Of The Use Case Tested

1. Get all building names:
   Mark: 0.
   Comment: This functionality was not implemented and we were unable to retrieve the data as a batch.

2. Get building by coordinates:
   Mark: 10.
   Comment: The function correctly returns all the information on a building based on the coordinates provided.

3. Get building by name:
   Mark: 10.
   Comment: The function correctly returns all the information on a building based on the name provided.

4. Get all room names:
   Mark: 0.
   Comment: This functionality was not implemented and we were unable to retrieve the room names as a batch.

5. Get location by room number:
   Mark: 10.
   Comment: The function correctly returns the location of a room by providing the building name and room.

6. Get routes:
   Mark: 8.
   Comment:
   The function mostly provides the correct route between a start and end point for a user.

7. Insert building:
   Mark: 10.
   Comment: The function correctly inserts a building(location) and all its details into the database.

8. Insert building room:
   Mark: 10.
   Comment: Correctly inserts a new room into an existing building of the database.

9. Update building coordinates:
   Mark: 10.
   Comment: The function correctly updates the coordinates of an existing building in the database.

10. Update building name:
    Mark: 10.
    Comment: Correctly updates an existing locations name(building name).

11. Update building room:
    Mark: 10.
    Comment: Correctly updates an existing room name inside a given

building.

12. Update building room coordinates:
Mark: 10.
Comment: This function correctly updates the coordinates of an existing room inside a building.

13. Update batch of building through external file:
Mark: 3.
Comment: This functionality only works when updating the original data file and doesn't take into account the user using a separate input file .

14. Remove building:
Mark: 10.
Comment: The function removes a building and all its dependencies correctly from the database.

15. Remove building room:
Mark: 10.
Comment: This function removes a room correctly from building it is contained in.

2 Non-Functional Requirements Tested

1. Maintainability:
   Mark: 10.
   Comment: Classes are separated well and allows easier access and retrieval to the object data.

2. Performance:
   Mark: 10.
   Comment: Recognized that Postgres is the best performing database for this system, thus has better response times.

3. Scalability:
   Mark: 10.
   Comment: Return methods allow for batch processing, returning a collection of objects at a time, as opposed to one a time.

4. Interoperability:
   Mark: 10.
   Comment:
   This module runs with Java, similar to the other teams, and thus allows interchangeability with the other modules or the interfaces it can communicate with.

5. Usability:
   Mark: 10.
   Comment: Enough class accessors and mutators have been provided to allow full use of the data objects.

6. Data Integrity and Security:
   Mark: 10.
   Comment: Communication between classes leave little room for problems with data communication, GIS also has little to no interaction with any sensitive information so security measures need not be so strenuous.

7. Transparency:
   Mark: 10.
   Comment: GIS module returns the locations and as much information relating to it, which makes the need for it to be as transparent as possible, which is delivered. Class objects provide access to all the fields and the methods themselves are named to clearly describe their function.

8. Documentation:
   Mark: 10.
   Comment: All methods are well documented.

9. Reliability:
   Mark: 9.
   Comment: Methods of this subsystem consistently return correct data, minus getting routes, which is - nevertheless - still mostly correct.

10. Availability:
    Mark: 10.
    Comment: Methods are easily accessible and maintain consistency with return values.

3 Improving On The Existing Test Output

3.0.1 Verification

We asked the question whether or not the behaviour of their code/component is what we expected. In most cases this was a yes as it did meet many of the expectations. It did however have some things that we expected, such as batch collection, that were not present.

3.0.2 Validation

We also asked if the right ”thing” is being built. This was an issue as they missed out some key points of what was expected from the project such as getting batch objects. However they were on the right track but were missing some critical parts that were requested.

4 Coverage And Efficiency

Mark: 3.
Comment: They has very little coverage of test cases overall. Not all functions were tested in the given Apptest.java file. There was no testing of multi-threaded cases and would cause code to break under these circumstances and return wrong values if an update and a get function were called at the same time. This would happen under real world circumstances where multiple users would request data simultaneously. Their test cases are inefficient as most functions were not tested at all. Their only test case in their test file was if the class compiled correctly. Also there was functionality that was missing such as the ability to gain arrays of data such as all the builds or classrooms.