Assignments Three and Four: Object-Oriented Software Design and Implementation (Group Work)

Robert S. Laramee and Max L. Wilson

March 8, 2011

Contents

1 Problem Statement and Overview

In these assignments, each group will design and build an application called a Data Visualizer, resembling that pictured in Figure 1. The data visualizer provides the same basic functionality of traditional spreadsheet software. A detailed list of the requirements specifications is given in Section 2 (Functional Specification).

The software design and implementation is broken down into three major phases:

1. **Assignment 3: Phase I Design:** Each group proposes an object-oriented design for the *entire* application. The design requirements are described in detail in Section 3.2.

2. **Assignment 3: Phase I Implementation:** Each group implements the features described in Section 3.3. They are a subset of the feature specification given in Section 2. The implementation follows the guidelines given in Section 6.

3. **Assignment 4: Phase II Hand-Off and Assessment:** Each group then hands their full design and partial implementation to another group. Correspondingly, each group takes over another groups' design and partial implementation. Each group then writes a short report assessing the project work they have taken over with respect to the quality of the design, implementation, and testing. A detailed description of this phase is provided in Section 4.2.

4. **Assignment 4: Phase II Implementation:** Each

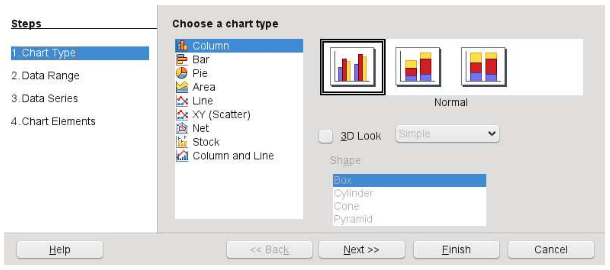


Figure 1: A screen-shot from an example Data Visualizer application each team is to design and implement.

group implements V2.0 of the system, i.e., the features described in Section 4.3. Again the implementation follows the guidelines given in Section 6.

5. **Phase III (Assignment 5) Implementation:** Assignment 5 is a re-engineering task where each individual implements features which are not yet defined. The implementation follows the rules given in Section 6. The full description of assignment 5 will be provided in a future document.

You and your team are building an application for a customer called *Bob*. And Bob is a representative of a company called *Bob Incorporated*. [1] Thus, any questions you have regarding the functionality of your system should be directed to customer Bob. Hint: Wake up Customer Interface Manager.

2 About the Client and the Assignments (Functional Specification)

The client requests the development of a data visualizer shared by many users at a typical university. Users are staff, students, and their relatives and friends. The data visualizer is located on an open access computer within the student center, and every university member can use the application with a unique user ID, which also acts as the password.

The data visualizer allows university members to visualize information about any data of their choosing. There are many types of visualizations, each of which has different characteristics. There are various types of users, each of whom may have different controls and views of the recorded data and visualizations.

[1]The Welsh branch of Bob Incorporated is called *Bob Limited*.

The overall goal of Assignments 3 and 4 is the delivery of the data visualizer to the client. This will be delivered in two phases, a design and small prototype (version 1) at the end of Assignment 3, and a completed prototype (version 2) at the end of Assignment 4.

Both assignments will be completed and assessed in groups. At the end of assignment 3, the version 1 developed by each group will be handed over to a different group for continuing development. The hand-over details will be given before the deadline. The contents to be handed-over include all required submissions for assignment 3.

In addition to Assignments 3 and 4, there will be an assignment 5 which is delivered individually. For assignment 5, the client requests a late change to the software. The details are unknown at this stage, as the client has no clue about this. Each student will develop his/her own version 3 by making changes to the version 2 developed by their group. A good design for version 1 and version 2 is very helpful to make the development of version 3 easy.

2.1 The Client's Functional Specification

Note that this specification is incomplete and sometimes vague as in real-world situations. Each group is required to complete the specification during the design phase of Assignment 3.

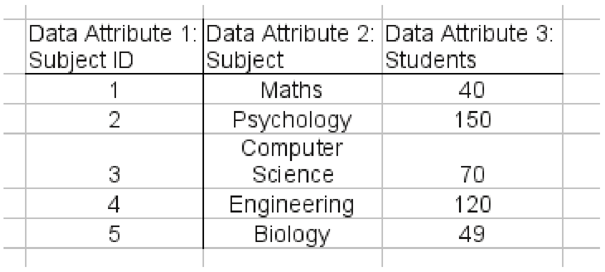
2.1.1 Data Types and Visualization: Terminology

The Data Visualization application supports the visualization of *multi-attribute data* as shown in Figure 2. Each column in the table corresponds to an *attribute* of the data, in other words a characteristic of each data item. The table in Figure 2 has three data attributes.

Each row of the table is called a *data record*. A data record contains the information corresponding to a single item, e.g., one course ID, one course name, and the number of students enrolled in that course.

Figure 2 shows just one simple example of a data set. The data visualization application supports any data set composed of records and columns, with arbitrary numbers of data items and corresponding attributes.

The data visualizer supports the storage and visualization of at least the following types of data including:



Data Attribute 1: Subject ID	Data Attribute 2: Subject	Data Attribute 3: Students
1	Maths	40
2	Psychology	150
3	Computer Science	70
4	Engineering	120
5	Biology	49

Figure 2: A partial screen-shot from an example Data Visualization application showing a data set as a Table.

1. integers, e.g., 1
2. floating point numbers, e.g., 1.00
3. letters, e.g., a
4. strings, e.g., alpha
5. boolean values

All numeric data types can be both positive and negative.

2.1.2 Types of Visualizations

The data visualizer supports at least the following types of visualizations, including:

1. A table view: The input data is simply presented in a table, 1 row for each record and 1 column for each data attribute, e.g., Figure 2.
2. Bar Chart: Each data sample of an attribute is mapped to a horizontal bar, the length of which represents the value. For multi-attribute data, each attribute gets a distinctive color.
3. Column Chart: Each data sample of an attribute is mapped to a vertical bar, the height of which represents the value. For multi-attribute data, each attribute is mapped to a distinctive color.
4. Pie Chart: Each data sample of an attribute is mapped to a pie slice, the angular width of which represents the value. For multi-attribute data, each attribute is mapped to a separate pie chart.
5. Line Chart: Each data sample is mapped to a point in space whose height represents the value. The points are then joined by a polyline. For multi-attribute data, each attribute is mapped to a separate polyline with a distinct color.
6. Bubble Chart: Each data sample is mapped to a circle whose radius represents the value. For multi-attribute data, each attribute is mapped to a separate set of circles with a distinct color.
7. Scatter Plot: Two data samples are mapped to an x-y point in space. Multi-attribute data is required here.
8. Other: any additional type of visualization of your choosing

Each group can add more visualizations. Links to ideas for more visualizations can be found on the CS_337 Data Visualization module web page which contains a link to the Many Eyes web site. Links to more sample data sets can be found on the InfoChimps web site: `http://infochimps.com/`.

Each type of visualization supports some user-options. The user-options for each visualization type may include but are not limited to:

1. Chart Title
2. Axis Label(s)
3. Axis Scale(s)
4. Minimum and Maximum axis values
5. Legend showing the correspondence between color and data attribute
6. The time and date on which the visualization was created
7. The author of the visualization.
8. A 1-2 sentence descriptive caption of the visualization
9. Resizing of the visualization
10. Saving the visualization to a GIF or PNG file

Each group can add more contents appropriate for a visualization type as they see fit.

2.1.3 Types of Users:

The data visualizer supports different types of university-based users, including:

1. Academic Staff
2. Non-Academic Staff
3. Student Members
4. Administrative Staff
5. University Alumni
6. Visitors and Guests

7. Other

Each type of user has different access controls to the data entered in the data visualizer. Some data can be entered (or visualized) by everyone, and other data may not. The right to edit or delete data or a visualization also depends on the type of user.

Each group can make an appropriate decision about the access control, as long as it's not the case that all users have the same access right.

Each type of user may also have a different default visualization.

3 Assignment 3 Tasks

For Assignment 3, your team provides a complete **design** for the entire system described in Section 2. It includes a reasonably complete class hierarchy. More detail about the design report is given in Section 3.2.

Your team also provides a **partial implementation** (V1.0) of the data visualizer as part of assignment 3. The features to be implemented are given in Section 3.3

3.1 Assignment 3 Deadlines: Electronic and Printed

Number of Credits: 25% of a 20 credit module
Recommended Hours: Approximately 50 hours (per group member)

Important Deadlines:

1. **6 December** by midnight: first copy of minutes of meeting
2. **Design Document (Section 3.2)**
   (a) ~~**11 February**~~ **18 February** by midnight: online submission of design document
   (b) ~~**14 February**~~ **21 February** by 14:00: paper/printed submission of design document
3. **Implementation (Section 3.3) and Group Report (Section 3.4.7)**
   (a) ~~**18 February**~~ **25 February** by midnight: online submission of Implementation Phase I software, doxygen output, and group report. Do not print out any source code or doxygen output on paper for the project. Digital copies of these files are stored online.
   (b) ~~**21 February**~~ **28 February** by 14:00: paper/printed submission of Group Report (only) for Phase I

Printed copies of required coursework documents are placed in the coursework submission drop-box or given to the Departmental Student Secretary in room 206 of Faraday Building.

3.2 Object-Oriented Design Report for Assignment 3 (Phase I)

The Design Report proposes an object-oriented design for the *entire* application. In other words, your team incorporates the full functional specification, both Phases I and II (Section 2) into the design. The Application Design Report is no more than 20 pages including text and diagrams and consists of the following sections:

3.2.1 Candidate Classes and Responsibilities:

Provide a list of candidate classes and their responsibilities. This list follows the class-card style described in the lectures and by Wirfs-Brock et al. [7, 8]. For each candidate class the team has identified, the following information is provided:

1. **Class Name:** (in bold)
2. Author:
3. SuperClass:
4. SubClasses:
5. Responsibilities: a list of services this class provides
6. Collaborations: a list of classes that this class may communicate with
7. Protocols: a list of methods that belong to this class including fully specified method signatures, i.e., method names, input parameters, and return parameters.
8. Unit Tests: some possible ways the class or object can be tested for functionality and robustness

3.2.2 Class Hierarchy Diagrams and Descriptions:

Each class hierarchy identified during your design process should be depicted in a diagram. The drawing style is to be modelled after the drawing style used in the lectures and by Wirfs-Brock [7, 8]. Your team is also welcome to use the drawing style of UML [2]. Each diagram is accompanied by a short description that describes the "is-kind-of" relationships used. Make sure that your team provides a justification that backs up its choices. Abstract classes are distinguishable from concrete classes in the diagrams. Your team should be able to identify three (or more) candidate class or object hierarchies.

3.2.3 Sub-systems Diagrams and Descriptions:

Each sub-system identified during your design process should be depicted in a diagram. Again, the drawing style is to be modelled after the drawing style used in the lectures (and by Wirfs-Brock [7, 8]) or the drawing style of UML [2]. Each diagram is accompanied by a short description that describes the "is-part-of" relationships used and depictions of classes that work closely together. Again, make sure that your team provides a justification for each sub-system, in other words, why you think a certain set of classes or objects belongs together in a sub-system. Your team should be able to identify two-three (or more) sub-systems.

3.2.4 Data File Format Description:

Describe the data file format used to store the data and all of their associated attributes. More information about the data is given in Section 2.

3.3 Object-Oriented Implementation for Assignment 3 (Phase I)

Must-Have Features: Version 1.0 for assignment three is a partial implementation. It includes (at least)

1. integer and floating point data types (from Section 2.1.1),
2. three types of visualization one of which must be the table view (from Section 2.1.2).
3. two user types (from Section 2.1.3),

3.4 Assignment 3 Submission: Design Report, Implementation, and Group Report

This section describes which files needs to be submitted for assignment 3 and how they should be submitted.

Assignments 3 and 4 are group work. Electronic copies of each report document, Java file, and JAR (Java Archive) file are kept online using the computer science web server. See the CS_254 module web page for instructions on how to place files on the computer science web server.

One member of each group, the **Planning and Quality Manager**, is required to create a folder called `cs254groupNa3`, where N is replaced with their group number, in their `public_html` folder. The `cs254groupNa3` online folder contains (and organizes) digital copies of all of the files that compose Assignment 3. In this way, anyone with a link to this folder can read and download all of the files that make up Assignment 3.

3.4.1 Minutes Protocol (5%)

A copy of your group minutes for three (or more) meetings held before the assignment deadline. Your minutes files should be called *"GroupNminutesDmonYr.txt"* where N is replaced by your group number, D is replaced by the calendar day your group met, Mon indicates the month that the group met, and Yr indicates the year, e.g., $Group1minutes23oct09.txt$. (You will be assessed on this.) Send a *link* to your first minutes to customer Bob as an email to reassure him that you are working hard on his product. A minimum of three minutes of meeting protocol is required and are submitted on behalf of the group by one of its members. *The first minutes has a special interim deadline before the other files.* See the given interim deadline. In addition, a digital copy of each minutes file is placed in a folder called `minutes` that resides in the `cs254groupNa3` folder described above.

3.4.2 Software Application Design Report (20%)

A copy of your Application Design Report online *and in printed form* is required by the deadlines indicated above. The file should be called *"GroupNdesignReport.pdf"* where N is replaced by your group number. PDF format is strongly encouraged. Open Office Document Format (.odt) is also acceptable. Word

document format (.doc or .docx) is not acceptable. See the following URL to convert word document format to PDF:
`http://cs.swan.ac.uk/~csbob/teaching/cs124-tutorial/`
Attention: you will be assessed on your conformity to the instructions. Don't forget to write the group member names and student numbers in the report. In addition, a digital copy of this report is placed on the CS web server in a folder called `design` that resides in the `cs254groupNa3` folder described above.

3.4.3 Application Implementation V1.0 (50%)

The application itself must also be submitted online. The main executable file should be called *"GroupNdataVisualizer.class"* where N is replaced by your group number. Pack your class files together in a Java Archive (.jar) file. Digital copies of implementation source files are placed on the CS web server in a folder called `implementation` that resides in the `cs254groupNa3` folder described above. Do not place a copy of every individual class file online, just the main executable and the Java archive file.

3.4.4 Application Implementation V1.0 Source Code (10%)

The application source code must also be submitted. The source file should be called *"GroupNsourceCode.zip"* where N is replaced by your group number. We recommend zipping the Java source code files together and storing them as one file online. You can also use tar to pack your source files together, e.g., you can log onto the CS server and type:

```
%> tar -cvf GroupNsourceCode.tar \
 implementation/*
```

Do not create a .rar file as these are not platform independent. Digital copies of all Java source files are placed on the CS web server in a folder called `implementation` that resides in the `cs254groupNa3` folder described above. The implementation follows the rules given in Section 6.

3.4.5 Application Implementation V1.0 Documentation (5%)

A copy of your application implementation documentation submitted online is required by the deadline(s) indicated above. Doxygen produces a series of linked HTML web pages in order to facilitate the browsing of project implementations. Refer to Section 7 for more on this topic. The HTML web pages produced by doxygen are placed on the CS web server in a folder called `doxygen` that resides in the `cs254groupNa3` folder described above.

3.4.6 Demo via Screen Capture (5%)

Each group uses screen capturing software to demonstrate the features of their application. Several links to (free) screen capturing software are given on the module web page.

The file(s) is named after the feature(s) being demonstrated e.g., `tableViewBarChart.mpg`. The movie files are saved in MPEG format. You may use as many screen capture files as necessary to capture the features of your application. One movie that captures all the features is ideal.

Your animated screen captures are also placed on the CS web server in a folder called `demo` that resides in the `cs254groupNa3` folder described above.

3.4.7 Group Report (5%)

A description of what and how each group member contributed to the project is submitted. More detail about the group report content is given in Section 8. The file is called *"GroupNmemberContributions.pdf"* where N is replaced by your group number. A digital copy of the group report is placed on the CS web server in a folder called `groupReport` that resides in the `cs254groupNa3` folder described above.

The group report also tells the reader: (1) where they can download the project files from (the URL) and (2) how to compile the code to produce the data visualizer application. A printed copy of the group report is also submitted.

3.4.8 Digital Copies of Files

Thus, when completing the submission of Assignment 3, the Planning and Quality Manager has a directory structure in their `public_html` folder that looks like this:

```
.../public_html/cs254groupNa3/
        demo/
        design/
        doxygen/
        groupReport/
        implementation/
        minutes/
```

Make sure that all of the files and folders are accessible (readable and executable) to anyone who has a link to your teams' `public_html` folder: e.g.,
`%> chmod -R ugo+rx *`

Points will be deducted for those submissions that do not follow the file naming conventions and required file formats.

Also, customer Bob may ask for a demonstration of the software by a group if he needs extra reassurance that the program is working as advertised.

4 Assignment 4 Tasks

Assignment 4 starts off with an evaluation phase of assignment 3. The assessment report is described in detail in Section 4.2.

4.1 Assignment 4 Deadlines: Electronic and Printed

Number of Credits: 25% of a 20 credit module
Recommended Hours: Approximately 50 hours (per group member)
Important Deadlines :

1. ~~**23 February**~~ **2 March** by midnight: copy of minutes of meeting
2. **Assessment Report (Section 4.2)**
   (a) ~~**25 February**~~ **4 March** by midnight: online submission of assessment of assignment 3 group work
   (b) ~~**28 February**~~ **7 March** by 14:00: paper/printed submission of assessment of assignment 3 group work
3. **Implementation (Section 4.3) and Group Report (Section 4.4.7)**
   (a) **18 March** by midnight: online submission of implementation Phase II software, doxygen output, and group report
   (b) **21 March 2010** by 14:00: paper/printed submission of Implementation Phase II group report

Paper/printed submissions can be dropped into the coursework submission drop-box or given to the Departmental Student Secretary in room 206 of Faraday Building.

4.2 Assessment Report for Phase II, Assignment 4

During the first week of Phase II, each group assesses the software design and implementation they have received and provides a report of their assessment. The report, no more than 10 pages, answers questions such as the following:

1. Have all the required files and documents been provided, e.g., design document, group report, minutes, source code, class files, and web pages produced by doxygen?
2. Have all the required files and documents been provided on time? If not, how late was the group in delivering the product?
3. Is the software design complete? Does it make sense? Have decisions been justified? Have at least three class hierarchies been identified and described? Have at least two-three sub-systems been identified and described?
4. Does the source code compile and provide a running application?
5. How closely does the source code conform to the coding conventions?
6. Do the functions specified in the group report actually work as advertised?
7. How well does the design match up with the implementation?
8. How robust is the software? What tests does it pass and when does it fail?
9. How does the software rate in terms of usability? Is it intuitive?
10. How complete is the doxygen output? Are both the short and detailed class descriptions there? Are both the class hierarchy and collaboration diagrams there? Is the source code there?

These questions are here only to give each team the ideas behind the assessment report (and to get you thinking about evaluation). More detail about the assessment will be given in semester two. Also, a sample copy of the assessment report will be given before the A3 implementation deadline such that each group knows how their application will be tested and evaluated. The team delivering a late product will lose points according to how many days after the deadline the requirements are fulfilled. Those lost points will then be awarded to the recipient team.

4.3 Assignment 4 Implementation

Assignment 4 implements the remaining features not specified above in Section 3.3. In other words, Assignment 4 implements everything described in Section 2 which describes the complete functional specification.

4.4 Assignment 4 Submission: Assessment Report, Implementation, and Group Report

Again, electronic copies of each report file, Java file, and JAR file are kept online using the computer science web server similar to Assignment 3.

The **Planning and Quality Manager** of each group creates a folder called `cs254groupNa4`, where N is replaced with their group number, in their `public_html` folder. The `cs254groupNa3` online folder contains (and organizes) digital copies of all of the files that compose Assignment 4.

4.4.1 Group Minutes (5%)

A copy of your group minutes for three (or more) meetings held before the assignment deadline is required. The protocol from Assignment 3 is used and the same file naming and format conventions are used as in Assignment 3. A minimum of three minutes of meeting protocol are required and are submitted on behalf of the group by one of its members. The first minutes has a special interim deadline before the other files. See the given interim deadline. In addition, a digital copy of each minutes file is placed in a folder called `minutes` that resides in the `cs254groupNa4` folder described above.

4.4.2 Software Assessment Report (10%)

A copy of your Software Assessment Report online *and in printed form* is required by the deadlines indicated above. More detail on this report is given in Section 4.2. The file should be called *"GroupNassessmentReport.pdf"* where N is replaced by your group number. PDF format is strongly encouraged. In addition, a digital copy of this report is placed on the CS web server in a folder called `assessment` that resides in the `cs254groupNa4` folder described above.

4.4.3 Application Implementation V2.0 (55%)

The next version of the application itself must also be submitted online. The main executable file should be called *"GroupNdataVisualizerV2.class"* where N is replaced by your group number. Pack your class files together into a Java Archive (.jar) file. Digital copy the class files are packed together as a JAR file and placed on the CS web server in a folder called `implementation` that resides in the `cs254groupNa4` folder described above.

4.4.4 Application Implementation V2 Source Code (10%)

The application source code must also be submitted. The source file should be called *"GroupNsourceCodeV2.zip"* where N is replaced by your group number. We recommend zipping the Java source code files together and storing them as one file online. You can also use tar to pack your source files together, e.g., you can log onto the CS server and type:

```
%> tar -cvf GroupNsourceCode.tar \
 implementation/*
```

Do not create a .rar file as these are not platform independent. Digital copies of all Java source files are placed on the CS web server in a folder called `implementation` that resides in the `cs254groupNa4` folder described above. Remember, the implementation follows the rules given in Section 6.

4.4.5 Application Implementation Documentation (10%)

The doxygen documentation submitted online is required by the deadline(s) indicated. Refer to Section 7, for more on this topic. The HTML web pages produced by doxygen are placed on the CS web server in a folder called `doxygen` that resides in the `cs254groupNa4` folder described above.

4.4.6 Demo via Screen Capture (5%)

Similar to assignment 3, each group uses screen capturing software to demonstrate the features of their application.

The file(s) is named after the feature(s) being demonstrated e.g., `barColumnPieChart.mpg`. The movie files are saved in MPEG format. You may use as many screen capture files as necessary to capture the features of your application, however, one movie file is ideal.

Your animated screen capture(s) is also placed on the CS web server in a folder called `demo` that resides in the `cs254groupNa4` folder described above.

4.4.7 Group Report (5%):

A description of what and how each group member contributed to the project is submitted. More detail about the group report is given in Section 8. The file is called *"GroupNmemberContributions.pdf"* where N is replaced by your group number. A digital copy of the group report is placed on the CS web server in a folder called `groupReport` that resides in the `cs254groupNa4` folder described above.

The group report also tells the reader: (1) where they can download the project files from (the URL) and (2) how to compile the code to produce version 2.0 of the application. A printed copy of the group report is also submitted.

4.4.8 Digital Copies of Files

Again, when completing the submission of Assignment4, the Planning and Quality Manager has a directory structure in their `public_html` folder that looks like this:

```
.../public_html/cs254groupNa4/
        assessment/
        demo/
        doxygen/
        groupReport/
        implementation/
        minutes/
```

Make sure that all of the files and folders are accessible (readable and executable) to anyone who has a link to your teams' `public_html` folder: e.g.,
`%> chmod -R ugo+rx *`

Customer Bob may ask for a demonstration of the software by a group if he needs extra reassurance that the program is working as advertised.

5 Assignment 5 Preview

Assignment 5 is a re-engineering task. The feature specification given in Section 2 will change. Each individual will then implement those changes. The better the teams are at designing and implementing assignments 3 and 4, the easier this task will be. Assignment 5 also encourages each group member to be highly involved in the group work in assignments 4 and 5. The more familiar you are with the previous work, the easier assignment 5 will be.

5.1 Assignment 5 Deadlines: Electronic and Printed

Number of Credits: 20% of a 20 credit module
Recommended Hours: Approximately 40 hours (individual work)
Important Deadlines :

1. **13 May** by midnight: online submission of A5 implementation and report
2. **16 May** by 14:00: paper/printed submission of A5 report

6 Object-Oriented Implementation for Assignments 3, 4 and 5

In parallel and after the design, the application also needs to be implemented in Java and Java Swing [1, 3]. Several helpful links to Java resources are given on the module web page.

The implementation needs to adhere to some rules. Customer Bob requires these rules because he wants a good product to be delivered. When he looks under the hood, he does not want to see a pile of scrap metal, but rather a finely tuned, carefully crafted machine.

1. **Coding Conventions:** Your team is required to follow *Bob's Concise Coding Conventions* [6]. See the module web page for a copy of this document.
2. **Code Comments:** Your team required to use Doxygen to comment your source code. Your code must be commented according to the guidelines given in *Bob's Concise Introduction to Doxygen* [5]. The doxygen program is free and available in the Linux lab of the computer science

department. See the module web page for a copy of Bob's Concise Introduction to Doxygen [5].

7 Implementation Documentation using Doxygen for Assignments 3, 4, and 5

Of course the actual implementation may change with respect to the original design. Thus, your team will also produce application implementation documentation as part of your project. The good news is that the application documentation is generated automatically using Doxygen. Use Doxygen to generate HTML web pages that provide:

1. A list of classes with a short description,
2. A list of classes with a detailed description including a list of all methods (both public and private) with the attributes defined in Bob's Concise Introduction to Doxygen [5],
3. Both class hierarchy diagrams and collaboration diagrams (generated automatically by Doxygen).
4. The source code of each class.

Points will be subtracted if any of these components is absent. Instructions on how to produce this output is provided by Bob's Concise Introduction to Doxygen [5] and `doxygen.org`.

8 Group Report for Assignments 3 and 4

Each group member is obliged to contribute two classes, both design and implementation, to assignments 3 and 4.

The group report indicates which group member(s) contributed to the project and how. The group report, no more than 5 pages, describes who contributed to:

1. classes in the design,
2. classes and sub-systems in the implementation,
3. testing: both unit testing and integration testing.

Each manager writes part of the report. Thus, there are sections written by the (1) Customer Interface Manager, (2) Design Manager, (3) Implementation Manager, (4) Test Manager, and (5) Planning and Quality Manager. Each section answers the questions that each manager is responsible for addressing. The questions are provided in the software engineering lecture on *Team Roles and Group Work* [4].

Every student needs programming practice. This is one of the main reasons we are here. The group report must contain a list of each group member along with the classes they implemented. Group members who have not implemented any classes will be penalized and may not receive any credit. Group members who are strong in implementation help those that are weaker by teaching them and giving them guidance on how to implement a class or write methods in Java.

The group report also lists which features were:

1. implemented and are working properly,
2. implemented and are not working properly,
3. not implemented with an explanation of why they were not implemented.

The report also also informs the reader if any unexpected problems arose during the course of the assignment. Likewise, the group experienced success in some areas, both expected and unexpected, this is included.

The group report also tells the reader: (1) where they can download the project files from (the URL) and (2) how to compile the code to produce the application.

9 Project Hints

1. Customer Bob recommends storing the data as human-readable ASCII or Unicode and is CSV (comma-separated values) format.
2. Remember that it is best to have a robust and stable application with fewer features than an unstable, full-of-bugs program with many features.
3. Test your application thoroughly. This includes unit testing, sub-system testing, and integration testing. Your program should be able to handle random input without crashing. Customer Bob is going to test your program with some very strange tests, e.g., negative letters and many

other strange things. If the program crashes, less money will be paid.

4. This is likely the most difficult and costly assignment you have undertaken so far in your degree (or life for that matter), so make sure to get started early.

5. All group members are expected to contribute to all phases of the development including design, implementation, and testing. Group members who are weak at programming may get help from other members who are better at it. The same is true for design and testing. The author fields in both the class design and implementation reflect individual group member contributions.

6. Allow your group extra time to learn how to use Doxygen.

7. More hints for the assignments will be given out in lectures.

8. Ask questions in your tutorial.

9. Why not use skype for some of your group meetings?

10. Do not use email to resolve conflict.

11. In order to get a distinction level score on assignments 3 and 4, you have to do very well or excellent for each component of the deliverables, e.g., (1) design, (2) implementation, (3) testing, (4) documentation. If any one of these aspects is weak, your project cannot receive a distinction level score.

10 Learning Outcomes and Transferable Skills

Learning Outcomes: Students will gain an understanding of the principles of software engineering; an understanding of the key HCI concepts in the context of system evaluation and design; the ability to design and evaluate GUIs; an understanding of object-oriented programming concepts, and knowledge of their applications in software design and engineering processes; the ability to build GUIs and skills of event-driven programming; experience and appreciation of group work; skills of project management.

Transferable Skills: Problem solving through analysis and abstract reasoning. The ability to read critically, to precis and judge information. Experience and appreciation of team work, time management, project management, and risk assessment. Skills in written communication and documentation. The ability to learn and use computer systems and software packages effectively.

References

[1] P.J. Deitel and H.M. Deitel. Java for Programmers. Prentice Hall, 2009.

[2] M. Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Object Technology Series. Addison-Wesley, third edition, September 2003.

[3] T. Gaddis and G. Muganda. Starting Out with Java, From Control Structures through Data Stru
Addison Wesley, first edition, 2007.

[4] W.S. Humphrey. TSP: Coaching Development Teams. Addison-Wesley, 2006.

[5] R. S. Laramee. Bob's Concise Introduction to Doxygen. Technical report, The Visual and Interactive Computing Group, Computer Science Department, Swansea University, Wales, UK, 2007. (available online).

[6] R.S. Laramee. Bob's Concise Coding Conventions (C^3). Advances in Computer Science and Engineering (ACSE), 4(1):23–26, 2010. (available online).

[7] R. Wirfs-Brock and A. McKean. Object Design: Roles, Responsibilities, and Collaborations. Addison-Wesley, 2003.

[8] R. Wirfs-Brock, B. Wilkerson, and L. Wiener. Designing Object-Oriented Software. Prentice-Hall, 1990.