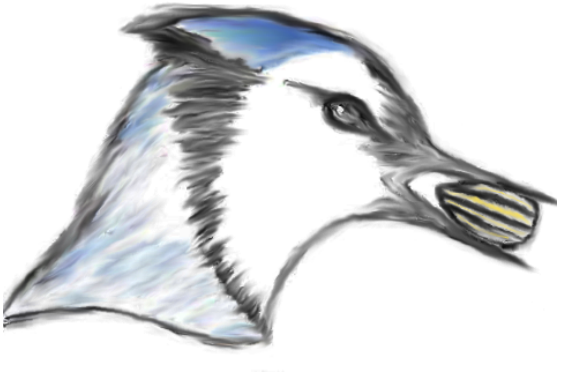


TEAM BIRDFEEDR
Project NoteBook

By

Josh Gillham
Wei Hang
Thomas Macari
Miguel Roman-Roman
Di Tran
Nathan Witt

Contents

Executive Summary

Team BirdFeedr is a small group of student software developers. Our members are Josh Gillham, Wei Huang, Thomas Macari, Miguel Roman, Di Tran, and Nate Witt.

Our mission as Team BirdFeedr is to inspire people to reach their computer science dreams by providing tools that help them overcome the barriers of programming.

Team BirdFeedr chose to develop an extension for the BlueJ IDE for their CS4260 class. We decided on our tools and to follow Agile methodologies that we previously learned in CS 4250 to tackle the contents of the extension. We held several team building events, communicated through email, voice chat, live person, and created group policies to effectively establish communication between members of the group.

We then explored BlueJs API by creating exploration projects. After the exploration projects, we conducted a survey and found that students had a difficult time understanding the syntax and nature of computer languages and programming. From this information, we decided on a BlueJ extension that would assist our users in understanding how to program.

The name of our extension is BlueJ TA. BlueJ TA's current features include: Being able to work on exercises using the BlueJ IDE, loading exercises from a local folder, and giving quick feedback on the correctness of the exercises.

Our product is by no means complete, but it contains a complete framework that future developers can build on. Our repository contains all the source files, as well as a build file so developers can immediately start working. Bundled with our product are some sample exercises so users can practice right away.

Overall, our time in Software Engineering Practices has proven fruitful, and this portfolio demonstrates that a team of 6 senior students, when working together, can produce a product that is not only demonstrable immediately, but also has enough forethought that when it will be passed on, it can be further developed into the product it was envisioned to be.

Communication

Summary Communication was essential to the success of our group. We practiced four communication methods regularly and they were meeting summaries, status updates, online team meetings, and regular internal email. Each of these methods supported our group in different ways. The meeting summaries helped us remember what progress we made during meetings. The status updates were sent to our professor and helped him understand the progress we made. The online team meetings facilitated rapid communication and were used to make decisions. Further, online meetings were used to pair program. For example, we used an online meeting to debug a cross platform bug that was stalling our extension on a Mac. All our communication methods were essential to our success.

Team Meeting Notes

Description This collection of team meeting notes were created at every team meeting. Scribe duties were assigned to Josh at the beginning, and then handed off to Di on April 5th, 2015. The notes changed format during this transition, from a stenographic format to meeting minutes format. Team meeting notes were taken during the course of weekly team meetings held during class on Mondays and Wednesdays from 2 PM to 4 PM and on Saturdays at 2 PM. Team meeting notes allowed team members to revisit and recall decisions and issues discussed in previous meetings. The following artifact consists of a sample of meeting notes from both the stenographic format and the meeting minutes format.

```
BEGIN File: /Team Documentation/Meeting Notes/note-20150121Team-Meeting.txt
```

```
2015-01-21 14:40:41
Tomas

Took principles from Nevada state
works for seria nevada corp.

--
Someone working on collision detection.



Le trying to get off cafene fix.
got off for two weks



Thomas -- built chat server with

Di double major in theater
  Good at writing documenting, presenting. Not good with small details.
```

***** Lines 20-69 were ommitted in this abridged version. *****

```
Jody -- COnfidence planner.

Choices Pro/cons.

BlueJ
  -Pro low risk, java
  -Cons low reward,

Jody -- Think about goals for next 5 days.

Deliver message;
 Group membership, 1st, 2nd choice.
 Tools requested.


Thomas -- Goals research extensions.

Moodle
  -pros learning
  -cons
2015-01-21 15:53:27
```

END File: /Team Documentation/Meeting Notes/note-20150121Team-Meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note-20150123Team-Meeting.txt

---

2015-01-23 17:01:31

What Should the ext do?

BlueJ able to compile PHP

Code support C++ ect.

Group Assignment:

Each person analyses possible project outcomes.

Miguel:
Clarify each project outcome.
Remove ambiguity.

***** Lines 20-96 were ommitted in this abridged version. *****

Di: agrees.
Miguel: Agrees.

Group Policy -- Track which you did each day if it applies to project.

The report policy b/c effective after redmines comes online.

Time tracking b/c active in the pen and paper sense.
Once redmine becomes online then the time tracking policy becomes effective.

Wei -- emails jody w/ team name, svn/redmine.

We met our meeting time.

Nathan: never used blueJ. Will play around with it.
2015-01-23 18:02:57

END File: /Team Documentation/Meeting Notes/note-20150123Team-Meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150126-Team-meeting.txt

---

2015-01-26 14:04:06

Established communication.
Team name.
Obtained Redmine and SVN
Research into extensions
Narrowed down project goal.

Policies: Notebook and logging.

Named Thomas the SVN manager.
2015-01-26 14:09:48

2015-01-26 14:21:52

Picking our extensions.

Redmine Workflow.
Where to put what.

***** Lines 20-61 were ommitted in this abridged version. *****

Wei: asks the other team for help.

Thomas: everyone can be more involved.

JP Loves the agile approach to get something done.
  Choose a small set of features like copy n paste detector.

Scoping the project down to what is doable.

Think of a small increment.

Like the idea of working in parrallel b/c you could learn more.

Di: Chirp.

Thomas: Git is on the table.
  Just commit.

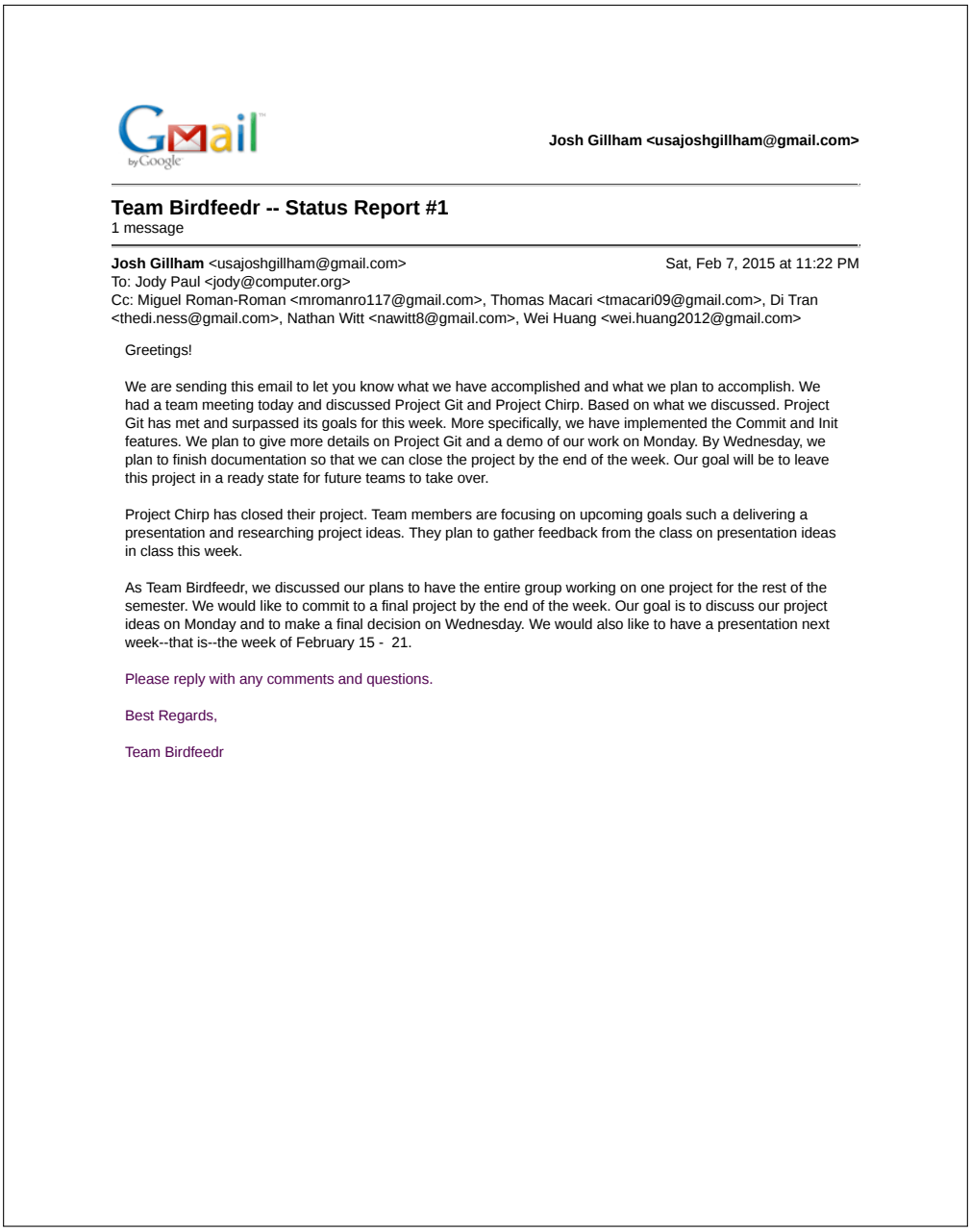
Weekly meetings on Saturday at 2 PM
 except Feb 23 and 28.
2015-01-26 15:52:42

END File: /Team Documentation/Meeting Notes/note20150126-Team-meeting.txt

Note: 36 artifacts were ommitted in this abridged version.

Status Reports

Description This collection of status reports sent to instructor and primary stakeholder, Dr. Jody Paul. Each status report contains weekly summarizations of team accomplishments and upcoming plans. These status reports were created because the team needed to communicated with the primary stakeholder in a consistent manner. By creating these status reports, an additional line of communication outside of class time was created between the team and the primary stakeholder.



Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Team Birdfeedr -- Status Report #1
1 message

Josh Gillham <usajoshgillham@gmail.com> Sat, Feb 7, 2015 at 11:22 PM
To: Jody Paul <jody@computer.org>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Greetings!

We are sending this email to let you know what we have accomplished and what we plan to accomplish. We had a team meeting today and discussed Project Git and Project Chirp. Based on what we discussed. Project Git has met and surpassed its goals for this week. More specifically, we have implemented the Commit and Init features. We plan to give more details on Project Git and a demo of our work on Monday. By Wednesday, we plan to finish documentation so that we can close the project by the end of the week. Our goal will be to leave this project in a ready state for future teams to take over.

Project Chirp has closed their project. Team members are focusing on upcoming goals such a delivering a presentation and researching project ideas. They plan to gather feedback from the class on presentation ideas in class this week.

As Team Birdfeedr, we discussed our plans to have the entire group working on one project for the rest of the semester. We would like to commit to a final project by the end of the week. Our goal is to discuss our project ideas on Monday and to make a final decision on Wednesday. We would also like to have a presentation next week--that is--the week of February 15 - 21.

Please reply with any comments and questions.

Best Regards,

Team Birdfeedr

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Team Birdfeedr -- Status Report #2
2 messages

Josh Gillham <usajoshgillham@gmail.com> Mon, Feb 16, 2015 at 9:19 AM
To: Jody Paul <jody@acm.org>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Greetings Dr. Paul,

The purpose of this email is to inform you on the progress we made in our last meeting on 2/14/15. First, we created a vision statement. Our vision statement is: "To inspire people to reach their computer science dreams by providing tools that help them overcome the barriers to programming." Second, we talked about the issues that were dividing our team and brought unity back. Fourth, we created a policy about final decisions and a policy on how to control side topics. Fifth, we decided to conclude Git Project. Sixth, we decided to create a vision document to help our group stay on track.

Please let us know if you have any questions.

Best Regards,
Team Birdfeedr

Dr. Jody Paul <jody@acm.org> Mon, Feb 16, 2015 at 9:44 AM
Reply-To: jody@acm.org
To: Josh Gillham <usajoshgillham@gmail.com>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Sounds like quite a lot was accomplished. I am looking forward to learning more details. Cheers. --JP
[Quoted text hidden]

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Team Birdfeedr -- Status Report #3
1 message

Josh Gillham <usajoshgillham@gmail.com> Wed, Feb 25, 2015 at 10:24 AM
To: Jody Paul <jody@acm.org>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Greetings Dr. Paul!

We wanted to tell you what we have accomplished while you were away. While you were away, we prepared a presentation by rehearsing, setting the stage, and creating presentation artifacts such as scripts and power point slides. We also created a list of projects that we can work on under our new vision statement.

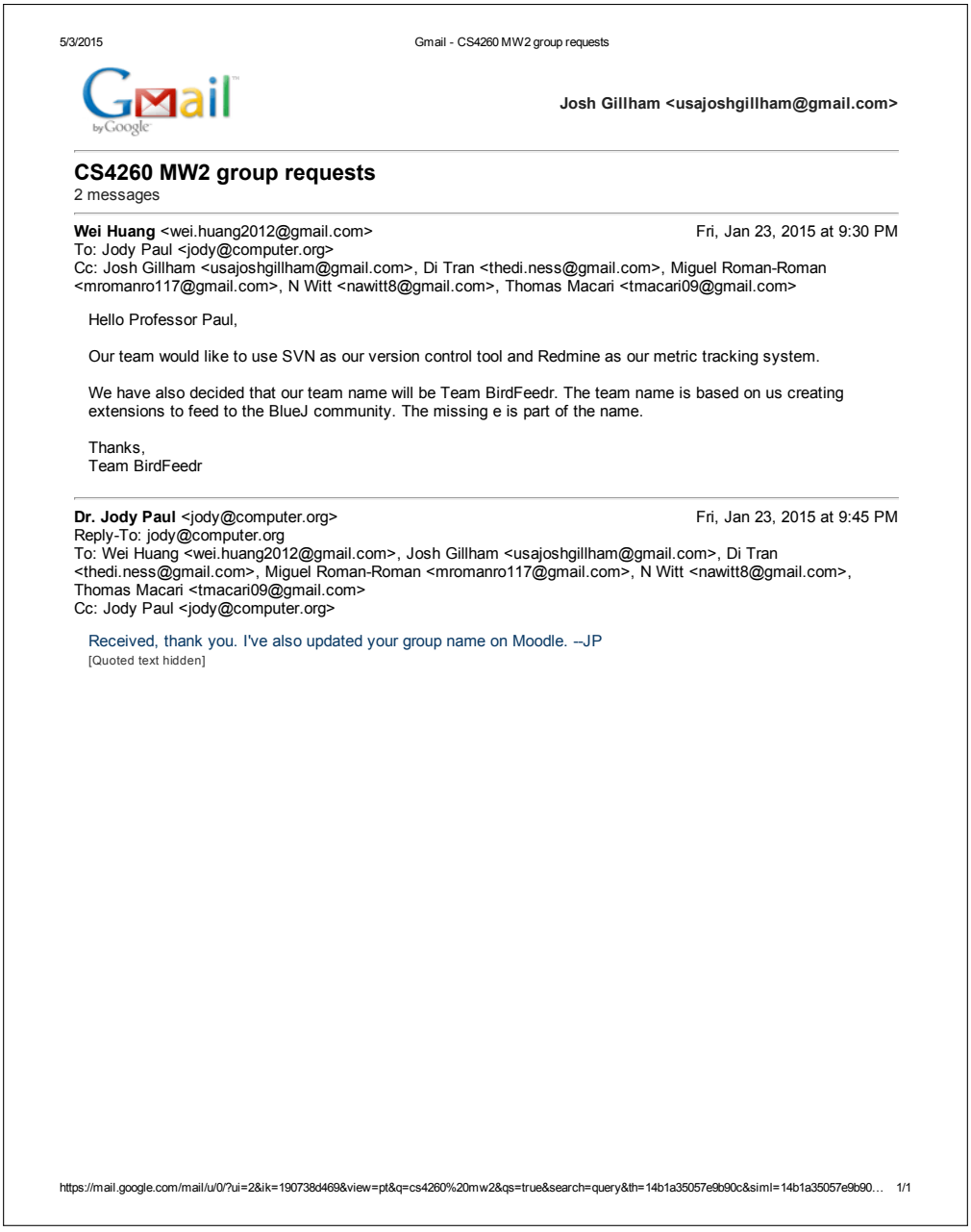
Tomorrow, we expect our presentation to take 15 minutes. After the presentation, we plan to answer any questions, reflect on the presentation comments, begin planning our next presentation, and select a project to work on. If you have any concerns or questions then please let us know.

Best Regards,
Team Birdfeedr

Note: 6 artifacts were ommitted in this abridged version.

Email Threads

Description The following artifact consists of screenshots of various email threads. Email was our teams main tool for communication while we were not together. By using emails, we were able to keep ourselves updated on the project and other events.



5/3/2015 Gmail - CS4260 MW2 group requests

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

CS4260 MW2 group requests
2 messages

Wei Huang <wei.huang2012@gmail.com> Fri, Jan 23, 2015 at 9:30 PM
To: Jody Paul <jody@computer.org>
Cc: Josh Gillham <usajoshgillham@gmail.com>, Di Tran <thedi.ness@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>

Hello Professor Paul,

Our team would like to use SVN as our version control tool and Redmine as our metric tracking system.

We have also decided that our team name will be Team BirdFeedr. The team name is based on us creating extensions to feed to the BlueJ community. The missing e is part of the name.

Thanks,
Team BirdFeedr

Dr. Jody Paul <jody@computer.org> Fri, Jan 23, 2015 at 9:45 PM
Reply-To: jody@computer.org
To: Wei Huang <wei.huang2012@gmail.com>, Josh Gillham <usajoshgillham@gmail.com>, Di Tran <thedi.ness@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>
Cc: Jody Paul <jody@computer.org>

Received, thank you. I've also updated your group name on Moodle. --JP
[Quoted text hidden]

https://mail.google.com/mail/u/0/?ui=2&ik=190738d469&view=pt&q=cs4260%20mw2&qs=true&search=query&th=14b1a35057e9b90c&siml=14b1a35057e9b90… 1/1

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Tomorrow's outcomes

6 messages

Josh Gillham <usajoshgillham@gmail.com> Tue, Jan 27, 2015 at 7:14 PM
To: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Greetings!

For tomorrow, since we have decided on our products, I am suggesting that we decide our **process models**. You all may want to take a look at the Distributed XP Process Model. Be warned! I have only read the abstract.

Since this course is about setting expectations and meeting them, as we go forward, we must **estimate the time to complete** the current stage of the process model and the **tangible outcomes**. As for the outcomes, they could be unit tests passed and/or documents produced.

We also need to set a time for our next meeting.

Please let me know if you have any comments or questions.

Best Regards,
Josh Gillham

Di Tran <thedi.ness@gmail.com> Tue, Jan 27, 2015 at 11:37 PM
To: Josh Gillham <usajoshgillham@gmail.com>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

I read the entirety of the DXP Process Model. I like it particularly because it addresses the shortcomings of a group that can't constantly meet face to face, but still adheres to a Agile process model. We should discuss its implementation tomorrow.

I'm in agreement regarding estimating how much time it'll take us to get stuff done, and the fit criterion / tangible outcomes associated with it.

As far as I'm aware, our next meeting is set for this Saturday at 2:00PM.

So on Wednesday (tomorrow), we want to cover:

- DXP Process Model / Other Models
- Time estimates
- Tangible outcomes

Is this correct?
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Wed, Jan 28, 2015 at 8:08 AM

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Food Decision

4 messages

Josh Gillham <usajoshgillham@gmail.com> Wed, Feb 4, 2015 at 3:56 PM
To: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Welcome to the food thread. Post your ideas about food and debate alternatives. TY

Wei Huang <wei.huang2012@gmail.com> Sat, Feb 7, 2015 at 3:10 PM
To: Josh Gillham <usajoshgillham@gmail.com>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Nathan Witt <nawitt8@gmail.com>

Btw the time we decided was 5:30pm
[Quoted text hidden]

Di Tran <thedi.ness@gmail.com> Sun, Feb 8, 2015 at 9:39 PM
To: Wei Huang <wei.huang2012@gmail.com>
Cc: Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Nathan Witt <nawitt8@gmail.com>

Can I put a vote in for the British Bulldog?
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Sun, Feb 8, 2015 at 9:41 PM
To: Di Tran <thedi.ness@gmail.com>
Cc: N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

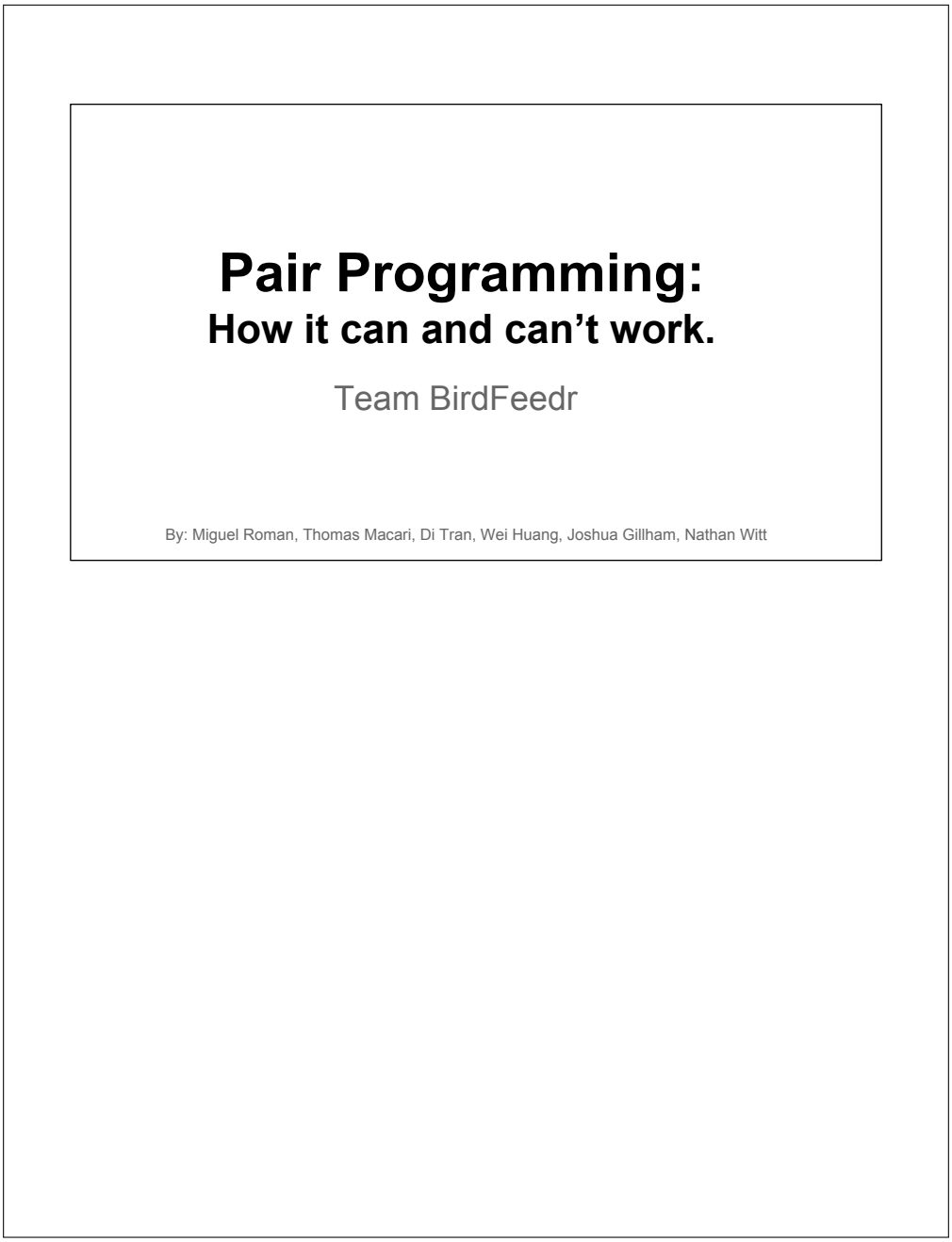
Sounds great!

[Quoted text hidden]

Note: 10 artifacts were ommitted in this abridged version.

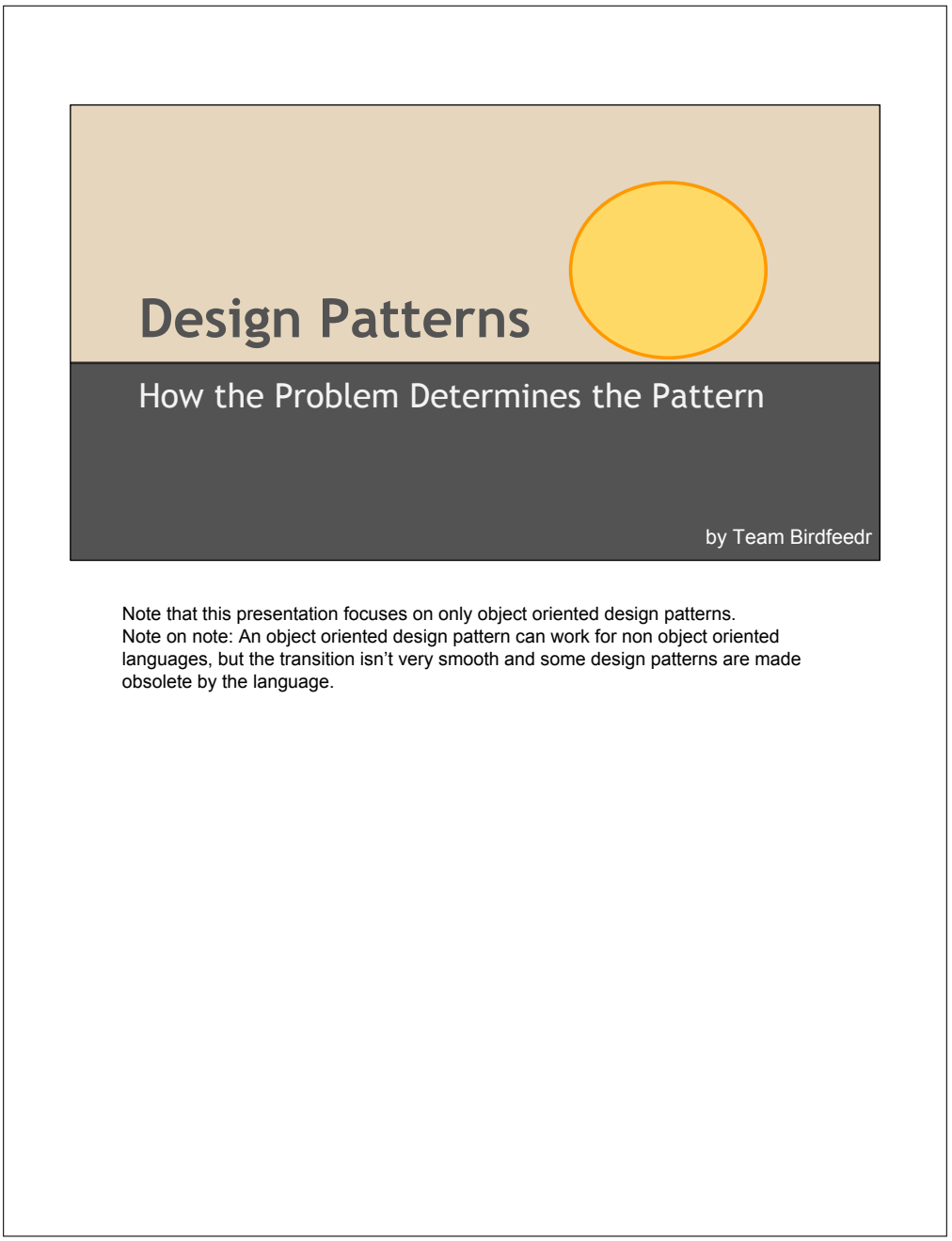
Presentation 1 Explains Pair Programming

Description Presentation one talked about our experience with pair programming. We started the presentation by defining pari programming. Next, we used our experiences to talk about different pair programming scenarios. Specifically, we created a skit to show good and bad programming pairs of three types: expert-expert, expert-novice, and novice-novice. We have included slides(s) from the presentation below.



Presentation 2 Explains Design Patterns

Description Presentation 2 talks about design patterns. First, we briefly defined design patterns. Second, we describe the elements of a design pattern. Third, we describe the types of design patterns. Fourth, we explain the need to identify the problem before prescribing a design pattern. Fifth, we compare design patterns to math formulas. Six, we show a real computer science application of design patterns. Seventh, we show how we made a mistake in selecting a design pattern because we did not properly identify the problem. Finally, we wrap up the presentation by emphasizing the need to identify the problem.



Presentation 3 Explains Vision Statements

Description Presentation 3 talks about our personal experience with vision statements. First, we share how vision statements may be used to handle conflicts. Last, we share how they can be used direct the team and unify the team. A sample of the presentation slides are included.

Final Presentation Shows Our Process and Product

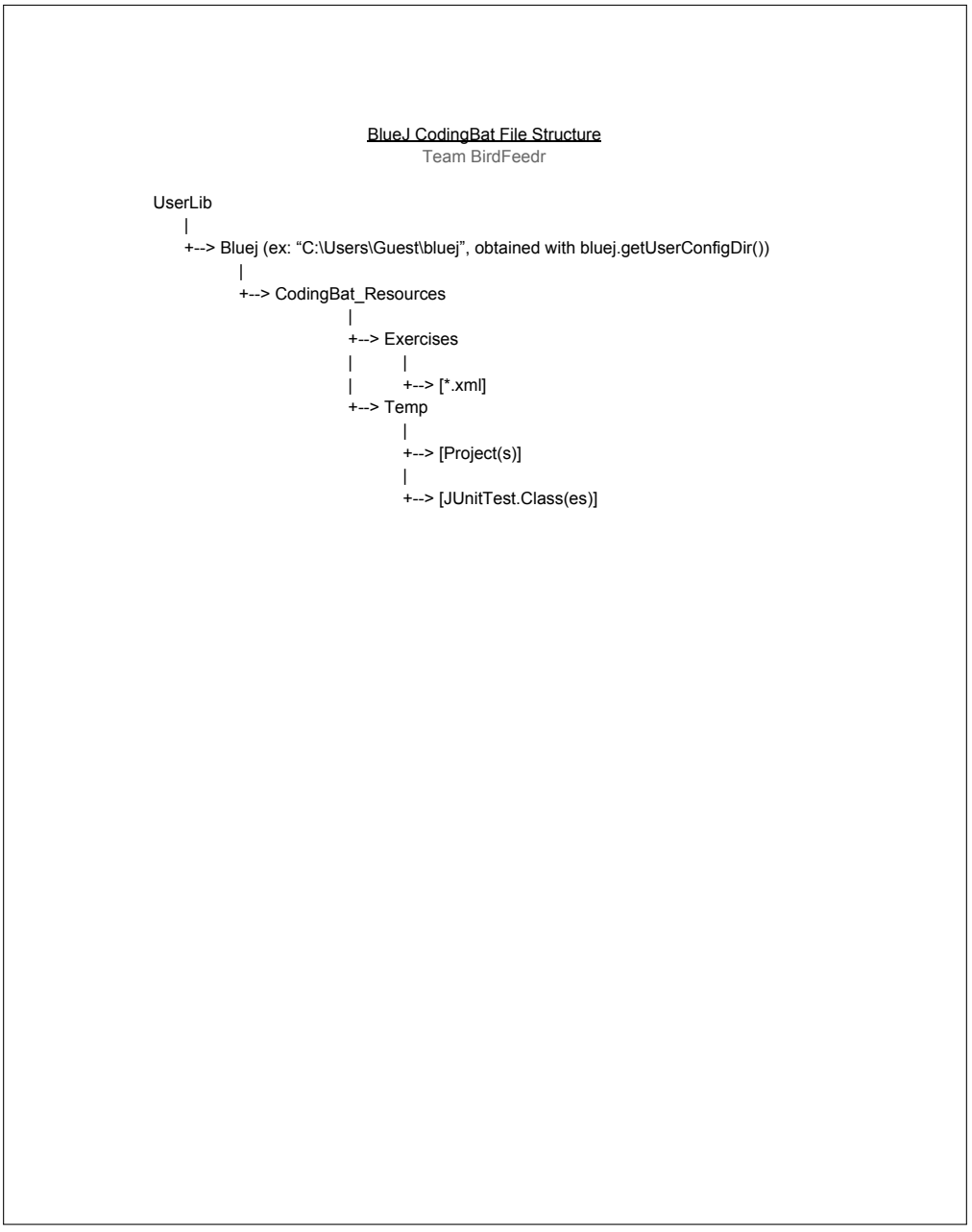
Description The final presentation talks about our work throughout the semester. First, we talk about the process we went through to develope the product. Last, we talk about the product by demoing the extension and explaining what happens behind the scenes.

Design

Summary Throughout our project we created multiple evolving designs about various aspects of the project. We first made designs that addressed how the main project could be split into modules that we then could work on simultaneously. Then we worked on more detailed designs that addressed more specific functions of those modules. As the project's requirements evolved and unforeseen bugs appeared, our designs changed to address these changes. In the following section we have shown some of our various key designs to the project. Some designs represent how a feature or function currently works in BlueJ TA. Other designs show how features worked in earlier versions depicting the evolution of BlueJ TA.

BlueJ TA Backend File Flow Proposals

Description The following artifact is a collection of proposals created during a design phase where we aimed to answer an important design question. The question we aimed to answer was What happens after the user selects and loads an exercise? To answer this question, our team created the following diagrams of the proposed workflow. These diagrams also helped describe a proposed file location for the exercises.



```
BEGIN File:
/Sandbox/Nate/BackEnd_FileFlow_Proposals/FileStructure/FileStructure_Example/Blu
eJCodingBat_Resources/Exercises/ExerciseA.xml
```

```
END File:
/Sandbox/Nate/BackEnd_FileFlow_Proposals/FileStructure/FileStructure_Example/Blu
eJCodingBat_Resources/Exercises/ExerciseA.xml
```

```
BEGIN File:
/Sandbox/Nate/BackEnd_FileFlow_Proposals/FileStructure/FileStructure_Example/Blu
eJCodingBat_Resources/Temp/UserProject/Adder.java
```

```
public class Adder {

    public int add(int a, int b){
        return a + b;
    }

}
```

```
END File:
/Sandbox/Nate/BackEnd_FileFlow_Proposals/FileStructure/FileStructure_Example/Blu
eJCodingBat_Resources/Temp/UserProject/Adder.java
```

Note: 8 artifacts were ommitted in this abridged version.

Proposed BlueJ TA Design for first Iteration

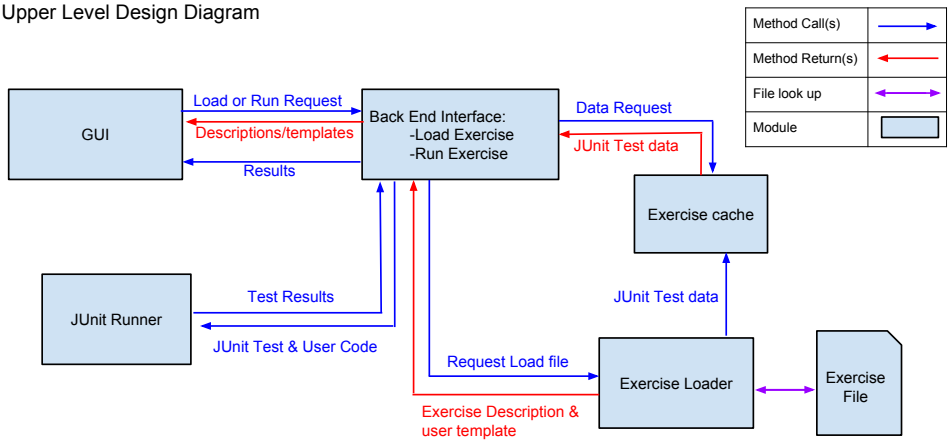
Description Design proposal contains upper level design diagram of how modules will link and communicate with each other on an upper level. We created this diagram to help us better understand how the extension will work and what needed to be done to create the first iteration as well as how we could also split up the modules to work independently and in parallel with one another. The following artifact is the document containing a flow diagram of the modules and user stories being addressed by this first iteration.

BlueJ Codingbat Design iteration 1

3/12/2015

By: Team BirdFeedr

Upper Level Design Diagram



User Story Flows

Running an exercise successfully:

A. GUI request the exercise to be run and passes the users encompassed method/code to the Back end interface.
B. The back end interface retrieves the JUnit test within the Exercise cache and passes both the Test and methods onto the JUnit Runner.
C. The Junit runner splices and compiles the code and returns a now running notification to the back end(which in turn returns to the GUI)
D. The Junit runner then runs the Junit test with the users code and sends the Results to the Back end interface when finished.
E. The back end Interface passes the Results to the GUI.
F. The gui interprets the Results and updates its display accordingly

Running an exercise Unsuccessfully:

A. GUI request the exercise to be run and passes the users encompassed method/code to the Back end interface.
B. The back end interface retrieves the JUnit test within the Exercise cache and passes both the Test and methods onto the JUnit Runner.
C. The Junit Runner splices, and attempts to compile the code but gets an error.
D. The Junit Runner returns the error to the interface.
E. The back end Interface passes the msg to the GUI.
F. The gui interprets the error msg and updates its display accordingly

Loading an exercise Successfully:

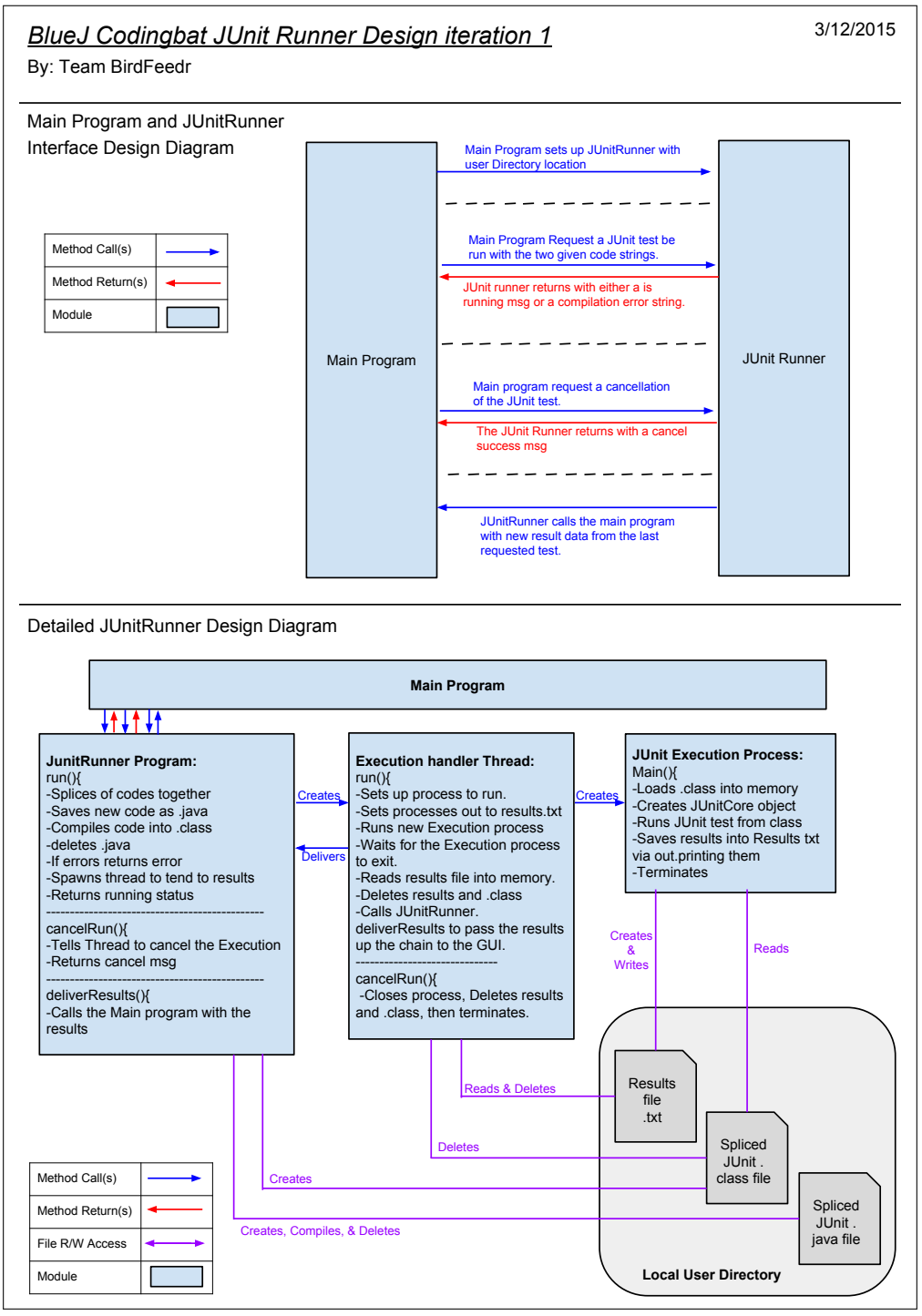
A. GUI request Excersize located at file "F" to be loaded from the interface.
B. Interface passes the request to the Exercise loader, and waits for a response.
C. The exercise loader locates the file and parses its contents into memory.
D. The Exercise parser stores the JUnit data in the Exercise cash then returns the Exercise description and User template to the interface.
E. The user interface passes the Exercise description and User template on to the GUI to be displayed on the screen.

Loading an exercise Unsuccessfully:

A. GUI request Excersize located at file "F" to be loaded from the interface.
B. Interface passes the request to the Exercise loader, and waits for a response.
C. The exercise loader fails to locate the file and returns an error message.
D. The user interface passes the error on to the GUI to be displayed on the screen.

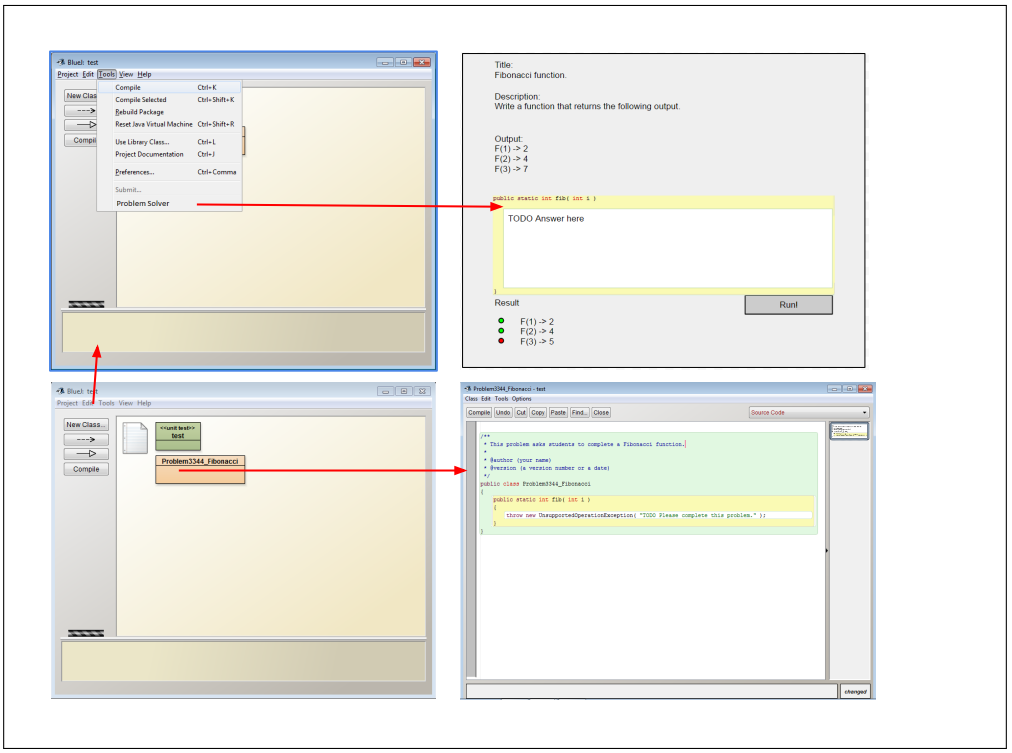
Proposed JUnitRunner Design v1

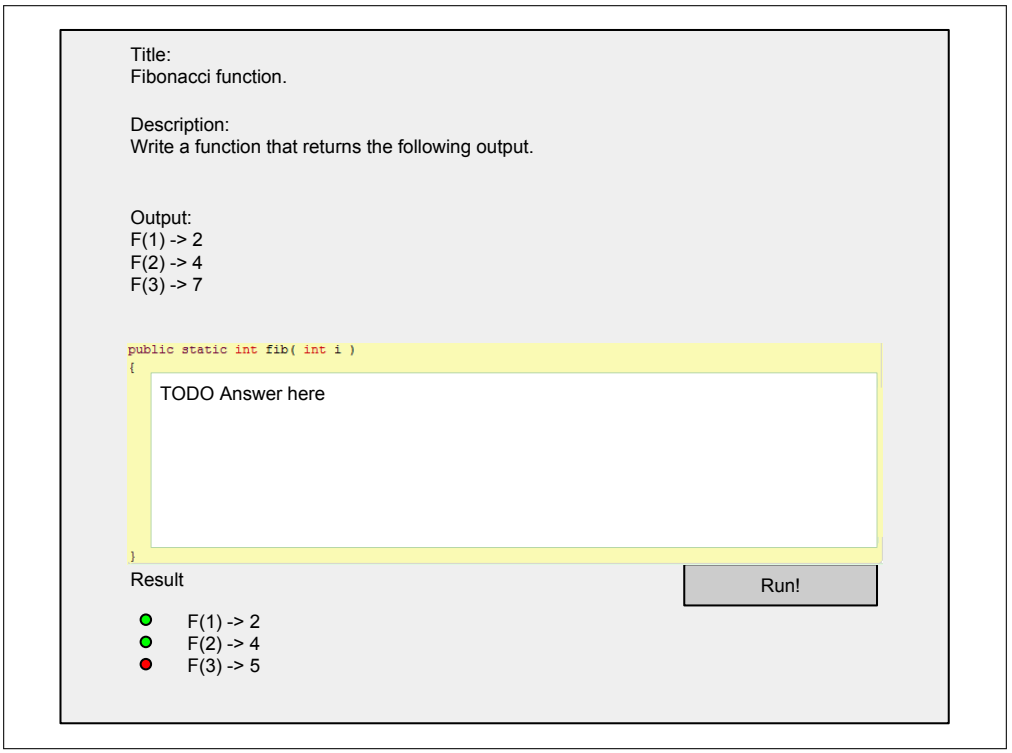
Description Design proposal containing possible workflow between the main extension and the JUnitRunner module. The diagram also includes class cards and detailed outlines of method calls, returns and read/write access between modules. The diagram helped the team understand how the JUnit runner would eventually be connected to the main program and get us ready and prepared The following artifact is a diagram detailing how the JUnit module would work class by class and how it will fit into the main project.

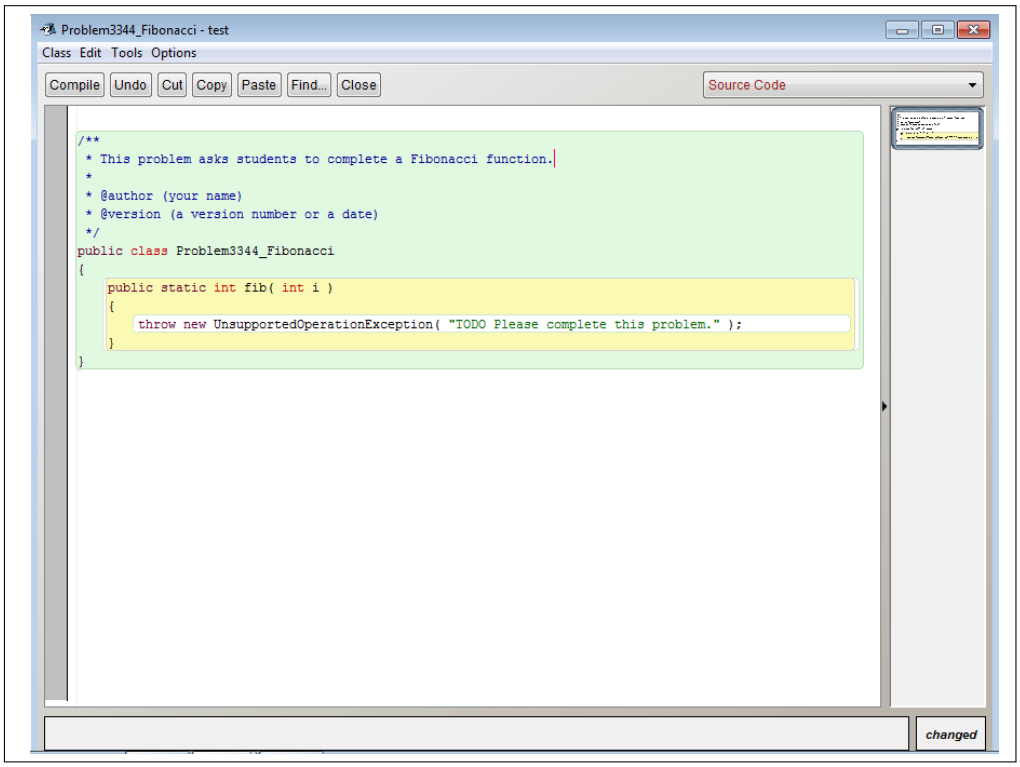


Proposed BlueJ TA Design for Graphical User Interfaces

Description Our team spent time at the board designing the GUI. We created this design to cover the flow of the extension, how it would look, and functionality. These designs also helped us all have a unified idea of what the project was going to be at the end. The following artifact is what we created on the board while designing the GUI and flow of the extension.




Title:
Fibonacci function.
Description:
Write a function that returns the following output.
Output:
F(1) -> 2
F(2) -> 4
F(3) -> 7
public static int fib(int i)
{
TODO Answer here
}
Result
Run!
F(1) -> 2
F(2) -> 4
F(3) -> 5

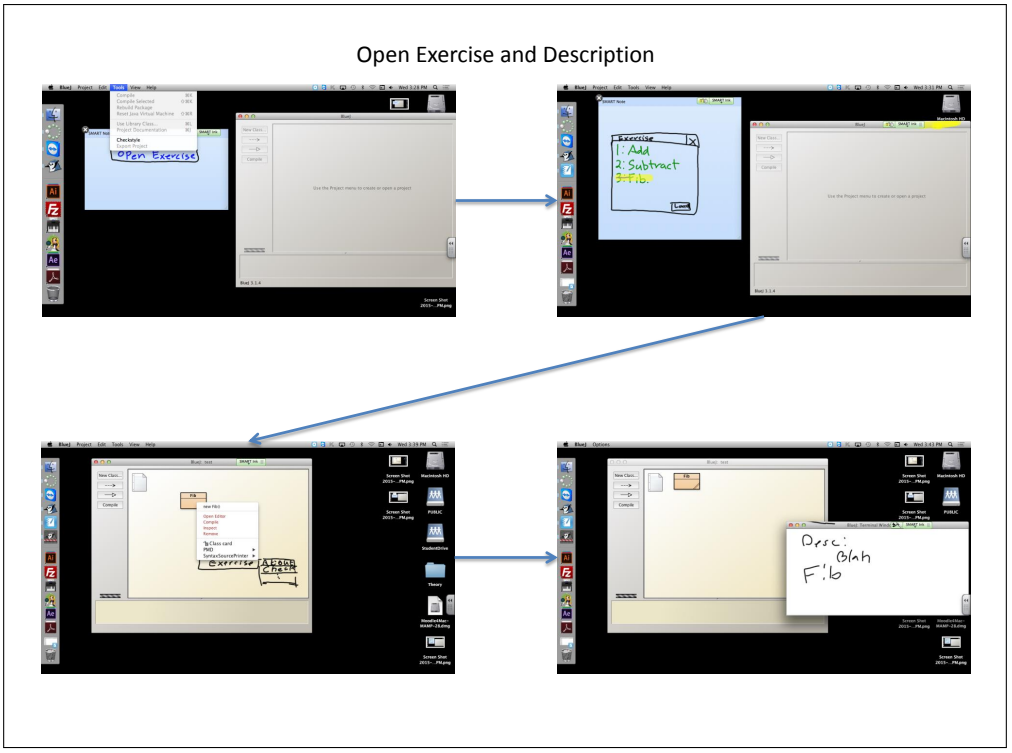


Note: 5 artifacts were ommitted in this abridged version.

BlueJ TA GUI Flow Designs

Description These diagrams where made to help in the resign in the GUI of BlueJ TA to accommodate a change in requirements. As well as to show how the user would navigate through the extension.

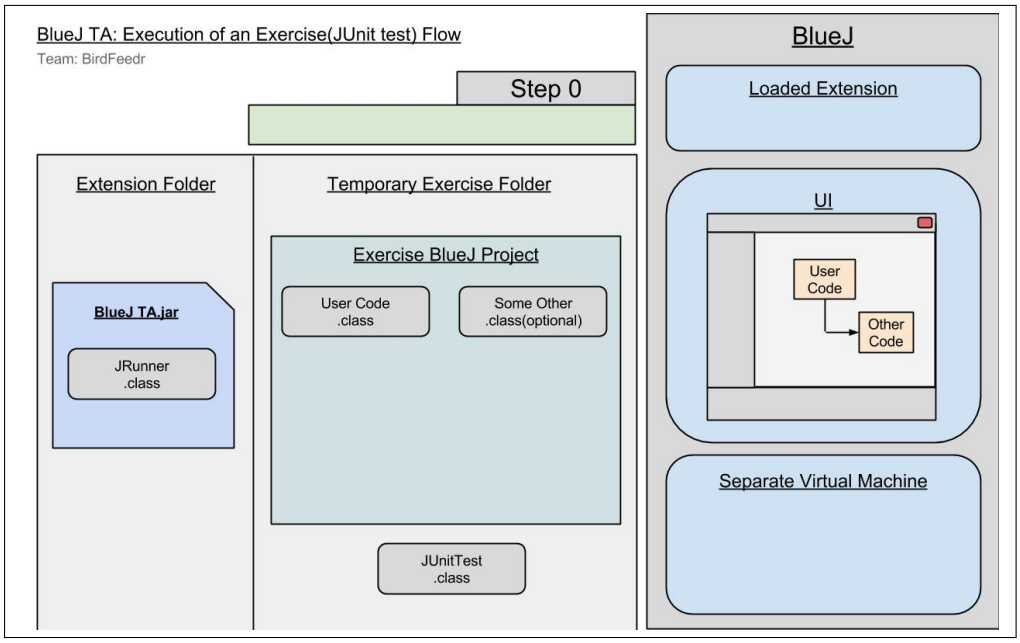
The following artifacts consists of the diagrams that describe the flow of the user in BlueJ TA.

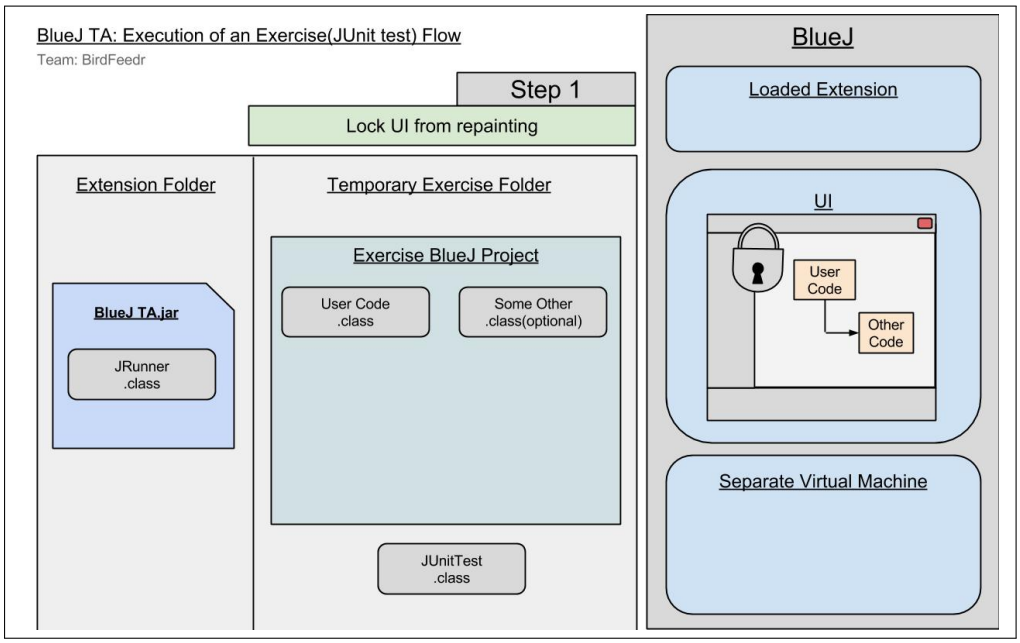


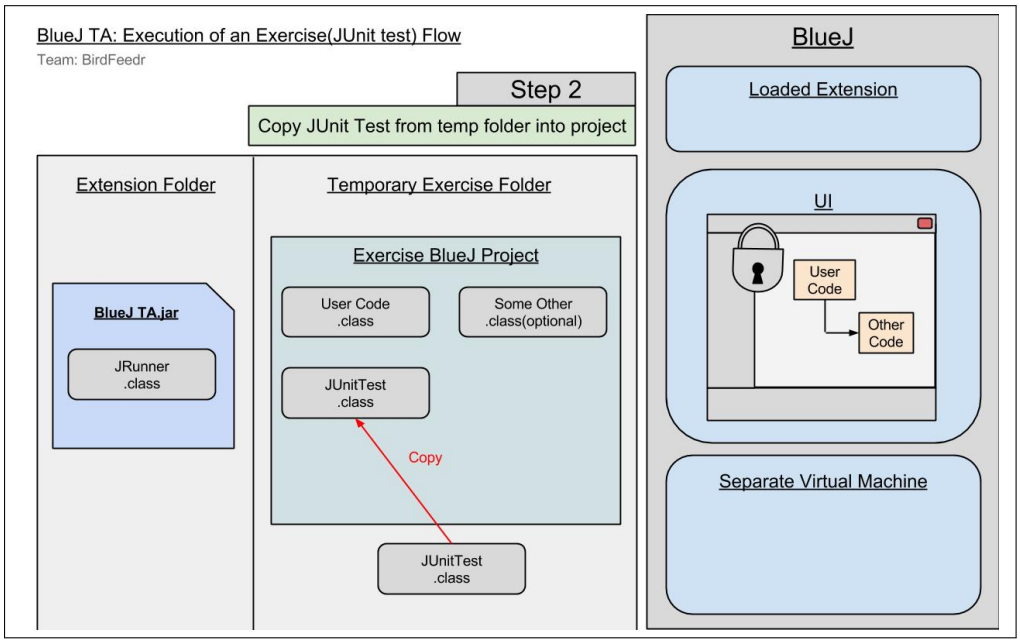
BlueJ TA Execution Design

Description These designs where made to help better describe how a exercise would be executed in BlueJ TA. Showing how the files and BlueJ's virtual machine is utilized to execute the test.

The following artifacts consists of the diagrams that describe the execution of an exercise.







Note: 7 artifacts were ommitted in this abridged version.

Documentation

Summary Documentation helps us communicate with future developers and remember key aspects about our project. We created documentation in a number of areas: a team vision, a project definition document, READMEs, source code Java Docs, etc. Our team vision describes the shared dream of our team. It was used to keep us going the same direction and to verify that our decisions were appropriate. Our project definition describes our project mission and what we were trying to accomplish. Our READMEs provide a number of features like explaining the directory structure of our project and how to install our extension. The source code documentation helps us remember how the code works, and also informs future developers should they choose to develop our project and learn about our code.

Project BlueJ TA Name

Description The name of the extension we made was called BlueJ TA because of the main function of the extension was to act as a teacher's assistant to help teacher teaches and students learn Java within BlueJ. To come up with this name we applied a naming heuristic to eliminate and reform various name ideas to decide the final product name.

The following artifacts consist of the meeting notes that where taken during the name meeting.

```
BEGIN File: /Team Documentation/Meeting Notes/note-20150123Team-Meeting.txt
```

```
2015-01-23 17:01:31

What Should the ext do?


BlueJ able to compile PHP

Code support C++ ect.


Group Assignment:

Each person analyses possible project outcomes.


---

Miguel:
Clarify each project outcome.
Remove ambiguity.
```

***** Lines 20-96 were ommitted in this abridged version. *****

```
Di: agrees.
Miguel: Agrees.


Group Policy -- Track which you did each day if it applies to project.


The report policy b/c effective after redmines comes online.

Time tracking b/c active in the pen and paper sense.
 Once redmine becomes online then the time tracking policy becomes effective.



Wei -- emails jody w/ team name, svn/redmine.


We met our meeting time.

Nathan: never used blueJ. Will play around with it.
2015-01-23 18:02:57
```

END File: /Team Documentation/Meeting Notes/note-20150123Team-Meeting.txt

Josh Gillham <usajoshgillham@gmail.com>

CS4260 MW2 group requests

2 messages

Wei Huang <wei.huang2012@gmail.com> Fri, Jan 23, 2015 at 9:30 PM
To: Jody Paul <jody@computer.org>
Cc: Josh Gillham <usajoshgillham@gmail.com>, Di Tran <thedi.ness@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>

Hello Professor Paul,

Our team would like to use SVN as our version control tool and Redmine as our metric tracking system.

We have also decided that our team name will be Team BirdFeedr. The team name is based on us creating extensions to feed to the BlueJ community. The missing e is part of the name.

Thanks,
Team BirdFeedr

Dr. Jody Paul <jody@computer.org> Fri, Jan 23, 2015 at 9:45 PM
Reply-To: jody@computer.org
To: Wei Huang <wei.huang2012@gmail.com>, Josh Gillham <usajoshgillham@gmail.com>, Di Tran <thedi.ness@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>
Cc: Jody Paul <jody@computer.org>

Received, thank you. I've also updated your group name on Moodle. --JP
[Quoted text hidden]

README's

Description Collection of READMEs dispersed at every level of the repository. These documents guide future developers with information like an explanation of the repository layout, install directions, and general repository usage recommendations. The following artifact consists of the READMEs created for project BlueJ TA.

```
BEGIN File: /Projects/BlueJ_TA/Code/README.txt
```

```
================================================================================
BlueJ Teacher's Assistant (TA) Extension
================================================================================
================================================================================




================================================================================
Description

The BlueJ TA Extension allows BlueJ users to practice Java in the BlueJ IDE.

================================================================================
Installing

 . Make sure that the JDK 1.7 or greater is installed.
 . Make sure that ANT is installed. If you don't have Ant then goto:
        http://ant.apache.org/ and click on "Binary Distributions"

================================================================================
```

***** Lines 20-61 were ommitted in this abridged version. *****

```
or <BLUEJ_HOME>\lib\extensions (Windows),
or <BLUEJ_HOME>/BlueJ.app/Contents/Resources/Java/extensions
(Mac, Control-click BlueJ.app and choose Show Package Contents)

For a single user for all projects.

<USER_HOME>/.bluej/extensions (Unix),
or <USER_HOME>\bluej\extensions (Windows),
or <USER_HOME>/Library/Preferences/org.bluej/extensions (Mac)

More information at: http://www.bluej.org/extensions/extensions.html

================================================================================
Authors

 - Josh Gillham
 - Wei Hang
 - Thomas Macari
 - Miguel Roman-Roman
 - Di Tran
 - Nathan Witt
```

```
END File: /Projects/BlueJ_TA/Code/README.txt
```

```
BEGIN File: /Projects/BlueJ_TA/Code/src/README.TXT
```

```
   ================================================================================
==
BlueJ Teacher's Assistant (TA) Extension
================================================================================
================================================================================



================================================================================
Directory Structure

src
+---+libs/                       Contains required libraries.
+---doc/                        Documentation. Generated by running "ant doc".
|   \---index.html               Documentaion main.
\---Extension/                   The main package.
    +---BackEnd/                 The package used to load and run exercises.
    |   \---runner/              This folder is hidden from BlueJ, but,
    |                             compiled by the build and used to run test
    |                             units on the user's solutions.
    +---exercises/               This folder contains the default exercise
    |                             files.
    \---GUI/                     This folder contains JavaFx GUI sources.

================================================================================
Authors

 - Josh Gillham
 - Wei Hang
 - Thomas Macari
 - Miguel Roman-Roman
 - Di Tran
 - Nathan Witt
```

```
END File: /Projects/BlueJ_TA/Code/src/README.TXT


BEGIN File: /Projects/Git/Git-Extension/README.TXT
```

```
  BlueJ Git Extension
================================================================================

Description
--------------------------------------------------------------------------------
The BlueJ Git Extension allows BlueJ users to integrate
 with [Git](http://git-scm.com/).

Getting started
--------------------------------------------------------------------------------
 . Create a Jar file and set the main class to "Git Extension" based
   in the manual section "Create (Executable) JAR Files"
   at http://www.bluej.org/doc/bluej-ref-manual.pdf
```

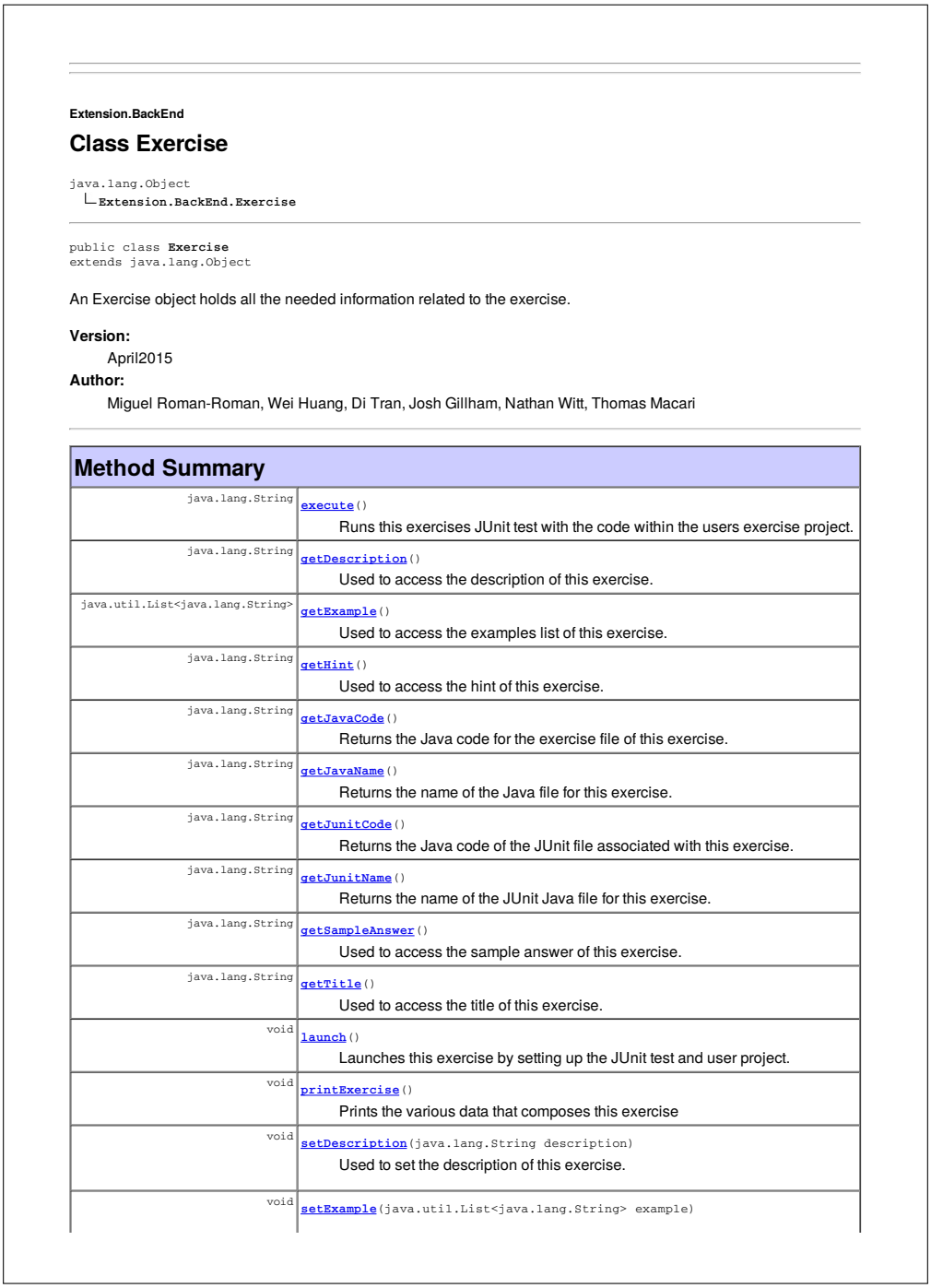
```
   Note: leave all options unchecked.

 . Install the extension based on the directions
    at http://www.bluej.org/extensions/extensions.html

Directory Structure
-------------------------------------------------------------------------------
Git-Extension
            +lib            --- Jar dependencies.
         A c t i o n s      --- Menu Actions.
        E x c e p t i o n s  --- Git Extension Exceptions.
         H e l p e r s      --- Simplifies talking to BlueJ and JGit.

Authors
-------------------------------------------------------------------------------
 - Miguel Roman-Roman
 - Josh Gillham
 - Thomas Macari

-------------------------------------------------------------------------------

END File: /Projects/Git/Git-Extension/README.TXT
```

Note: 1 artifacts were ommitted in this abridged version.

JavaDoc

Description The BlueJ TA project utilizes Java Doc to describe key aspects of the program and how it functions. Where Java doc is a standard documentation format to add documentation directly to the code of a program, describing classes methods and variables. We decided to add this documentation to our code to help increase the understandability of our project with the added compatibility of backed in documentation.

The following artifacts consists some samples of the Java doc for BlueJ TA.



Extension.BackEnd

Class Exercise

```
java.lang.Object
  └Extension.BackEnd.Exercise
```

```
public class Exercise
extends java.lang.Object
```

An Exercise object holds all the needed information related to the exercise.

Version:
April2015

Author:
Miguel Roman-Roman, Wei Huang, Di Tran, Josh Gillham, Nathan Witt, Thomas Macari

Method Summary

java.lang.String	execute() Runs this exercises JUnit test with the code within the users exercise project.
java.lang.String	getDescription() Used to access the description of this exercise.
java.util.List<java.lang.String>	getExample() Used to access the examples list of this exercise.
java.lang.String	getHint() Used to access the hint of this exercise.
java.lang.String	getJavaCode() Returns the Java code for the exercise file of this exercise.
java.lang.String	getJavaName() Returns the name of the Java file for this exercise.
java.lang.String	getJunitCode() Returns the Java code of the JUnit file associated with this exercise.
java.lang.String	getJunitName() Returns the name of the JUnit Java file for this exercise.
java.lang.String	getSampleAnswer() Used to access the sample answer of this exercise.
java.lang.String	getTitle() Used to access the title of this exercise.
void	launch() Launches this exercise by setting up the JUnit test and user project.
void	printExercise() Prints the various data that composes this exercise
void	setDescription(java.lang.String description) Used to set the description of this exercise.
void	setExample(java.util.List<java.lang.String> example)

Extension.BackEnd

Class FileUtil

```
java.lang.Object
  └ Extension.BackEnd.FileUtil
```

```
public abstract class FileUtil
extends java.lang.Object
```

Utility methods for dealing with files and BlueJ projects.

Version:
April2015

Author:
Miguel Roman-Roman, Wei Huang, Di Tran, Josh Gillham, Nathan Witt, Thomas Macari

Constructor Summary

FileUtil()

Method Summary

static void	closeProj(bluej.extensions.BlueJ bluej, java.lang.String name) Closes the any open bluej projects that have the given name.
static void	copy(java.io.File source, java.io.File dest) Copies the contents of the source file into the destination file.
static boolean	deleteDir(java.io.File path) Deletes the directory and all the contents within the directory.
static void	extractResource(java.lang.String resLoc, java.lang.String resName, java.io.File destDir) Saves the given resource within this jar to the a file in the directory specified.
static boolean	save(java.io.File path, java.lang.String contents) Saves the given contents into the given file specified by path.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

FileUtil

```
public FileUtil()
```

Method Detail

closeProj

Extension.BackEnd

Class RemoveThread

```
java.lang.Object
  └java.lang.Thread
      └Extension.BackEnd.RemoveThread
```

All Implemented Interfaces:

java.lang.Runnable

```
public class RemoveThread
extends java.lang.Thread
```

A thread used to clean up execution artifacts within a users project and unlock its ui once they are loaded into memory.

Author:

Team BirdFeedr

Nested Class Summary

Nested classes/interfaces inherited from class java.lang.Thread
java.lang.Thread.State, java.lang.Thread.UncaughtExceptionHandler

Field Summary

Fields inherited from class java.lang.Thread
MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY

Constructor Summary

RemoveThread(bluej.extensions.BClass testClass, bluej.extensions.BClass runClass, java.awt.Frame frame) Constructor

Method Summary

void	run() Remove the BClasses from the project and unlock the UI after the JRunners running flag has been set to true.

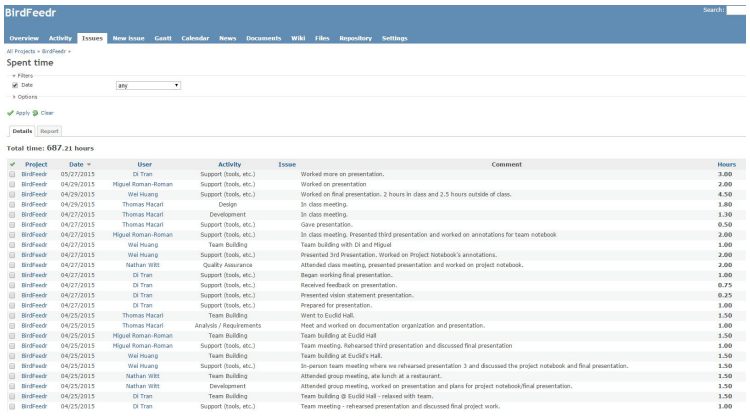
Methods inherited from class java.lang.Thread
activeCount, checkAccess, clone, countStackFrames, currentThread, destroy, dumpStack, enumerate, getAllStackTraces, getContextClassLoader, getDefaultUncaughtExceptionHandler, getId, getName, getPriority, getStackTrace, getState, getThreadGroup, getUncaughtExceptionHandler, holdsLock, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setDefaultUncaughtExceptionHandler, setName, setPriority, setUncaughtExceptionHandler, sleep, sleep, start, stop, stop, suspend, toString, yield

Methods inherited from class java.lang.Object
equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Note: 12 artifacts were ommitted in this abridged version.

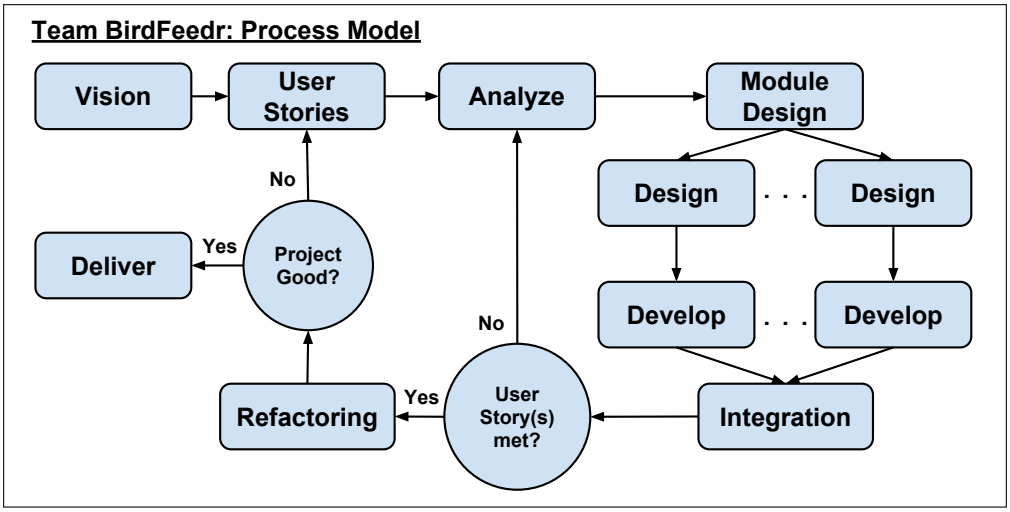
Time Log

Description The time log are the entries in our project management tool. The entries record our activies during the course of this semester.



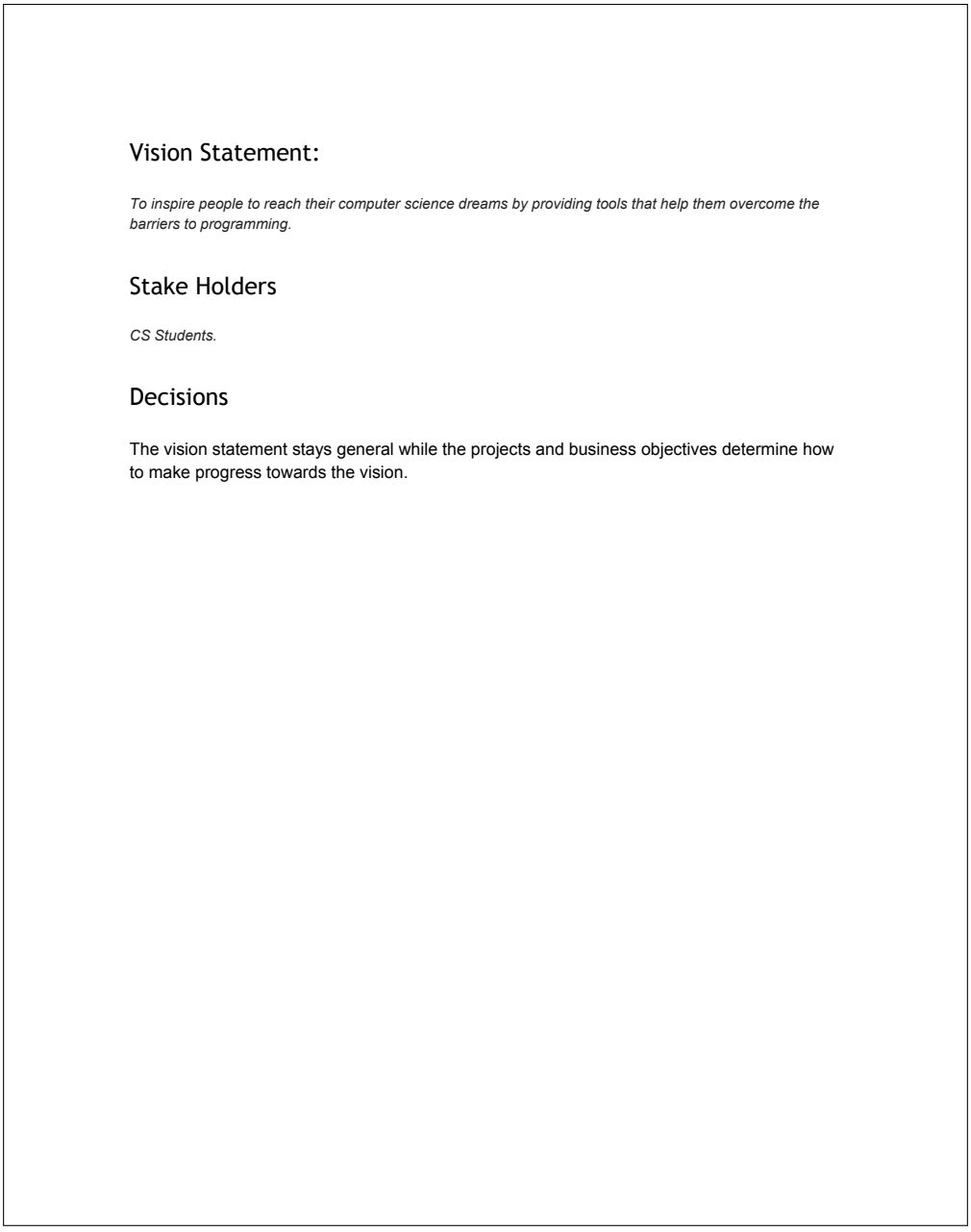
Process Model

Description This reflects the actual process we went through as a group. The diagram below shows the steps and their next steps.



Team Vision Statement

Description We crafted the following vision statement to reflect what we wanted to accomplish during this course. We used this statement to point out any priorities that were wrong. It also kept us going in the righ direction and unified our group.



Vision Statement:

To inspire people to reach their computer science dreams by providing tools that help them overcome the barriers to programming.

Stake Holders

CS Students.

Decisions

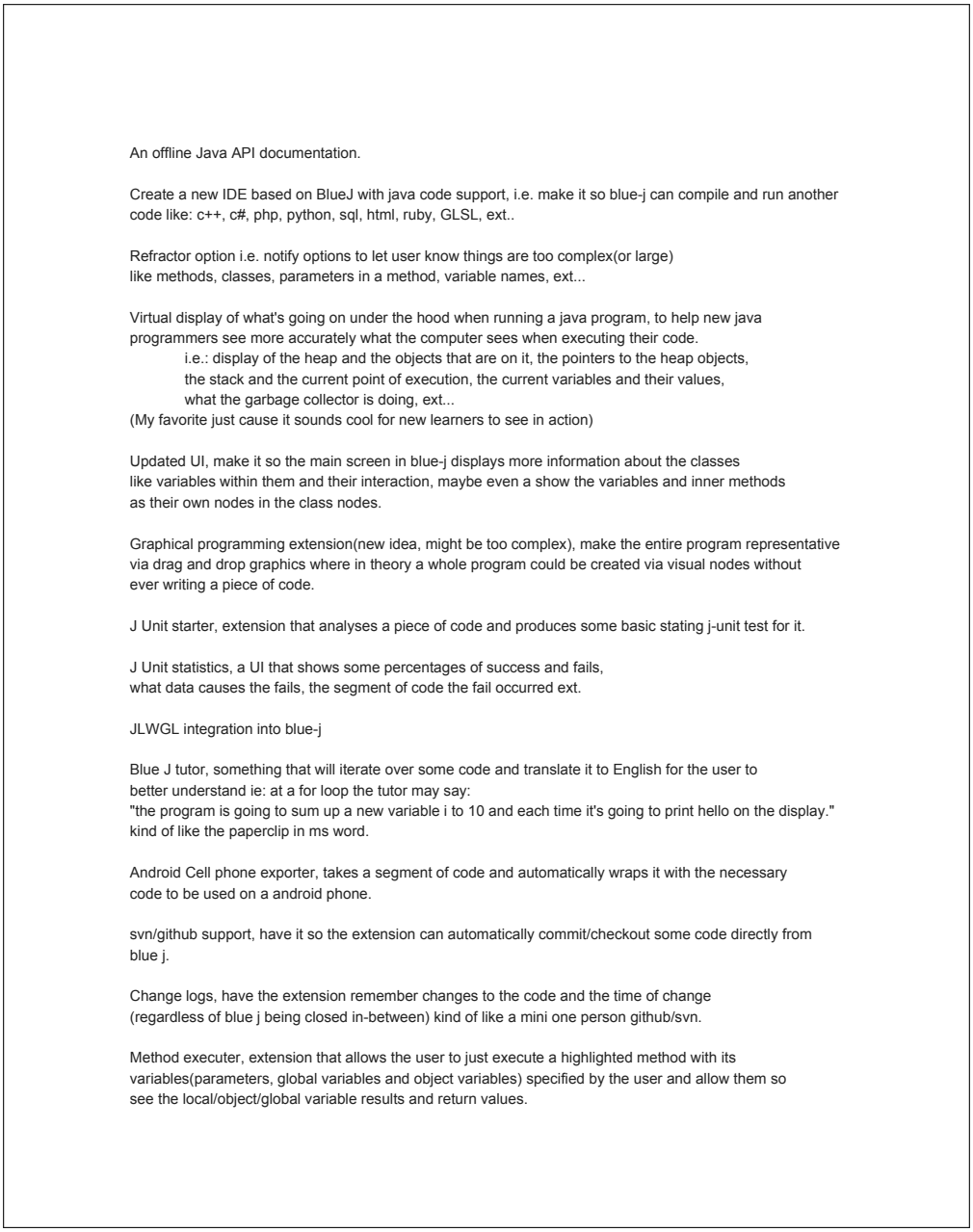
The vision statement stays general while the projects and business objectives determine how to make progress towards the vision.

Project License

Description We did a little research on the license and decided that we would like to use the Creative Commons license. We choose this because it aligned we our goals of making the product freely available and open source. We are not sure what the university policies are for content we create.

Final Project Ideas List

Description The following is the list of project ideas. As a group, we had to decide if any of these would be it.



An offline Java API documentation.

Create a new IDE based on BlueJ with java code support, i.e. make it so blue-j can compile and run another code like: c++, c#, php, python, sql, html, ruby, GLSL, ext..

Refractor option i.e. notify options to let user know things are too complex(or large)
like methods, classes, parameters in a method, variable names, ext...

Virtual display of what's going on under the hood when running a java program, to help new java programmers see more accurately what the computer sees when executing their code.
i.e.: display of the heap and the objects that are on it, the pointers to the heap objects, the stack and the current point of execution, the current variables and their values, what the garbage collector is doing, ext...
(My favorite just cause it sounds cool for new learners to see in action)

Updated UI, make it so the main screen in blue-j displays more information about the classes like variables within them and their interaction, maybe even a show the variables and inner methods as their own nodes in the class nodes.

Graphical programming extension(new idea, might be too complex), make the entire program representative via drag and drop graphics where in theory a whole program could be created via visual nodes without ever writing a piece of code.

J Unit starter, extension that analyses a piece of code and produces some basic stating j-unit test for it.

J Unit statistics, a UI that shows some percentages of success and fails,
what data causes the fails, the segment of code the fail occurred ext.

JLWGL integration into blue-j

Blue J tutor, something that will iterate over some code and translate it to English for the user to better understand ie: at a for loop the tutor may say:
"the program is going to sum up a new variable i to 10 and each time it's going to print hello on the display."
kind of like the paperclip in ms word.

Android Cell phone exporter, takes a segment of code and automatically wraps it with the necessary code to be used on a android phone.

svn/github support, have it so the extension can automatically commit/checkout some code directly from blue j.

Change logs, have the extension remember changes to the code and the time of change (regardless of blue j being closed in-between) kind of like a mini one person github/svn.

Method executer, extension that allows the user to just execute a highlighted method with its variables(parameters, global variables and object variables) specified by the user and allow them so see the local/object/global variable results and return values.

Team Policies Set Team Expectations

Description As a team we created a policy document to help set group expectations and manage group behaviour. The policies we made where subject to change over the semester should we find a need for a new policy or alteration of a faulty policy.

The following artifact is a copy of our team policy document.

```
BEGIN File: /Team Documentation/Birdfeedr Policies.txt
```

```
BIRDFEEDR POLICIES FAQ:

1. EMAILS
**********
1.1 INTERNAL EMAILS
- - - - - - -
Who should I send an email to?
      All group members, unless there is a reason not to
      (for example, if you want to send an email to your subgroup)
Somebody just sent an email to me, when should I reply?
      Within 24 hours.

EXTERNAL EMAILS
- - - - - - -
How should I send an email to Jody or anybody else?
      Send a draft to everyone else in the group, and specify
      a time in which people can suggest edits before sending.
How long should I wait before sending it?
      Wait until the end of the day before sending it.
I just got a draft, what should I do?
```

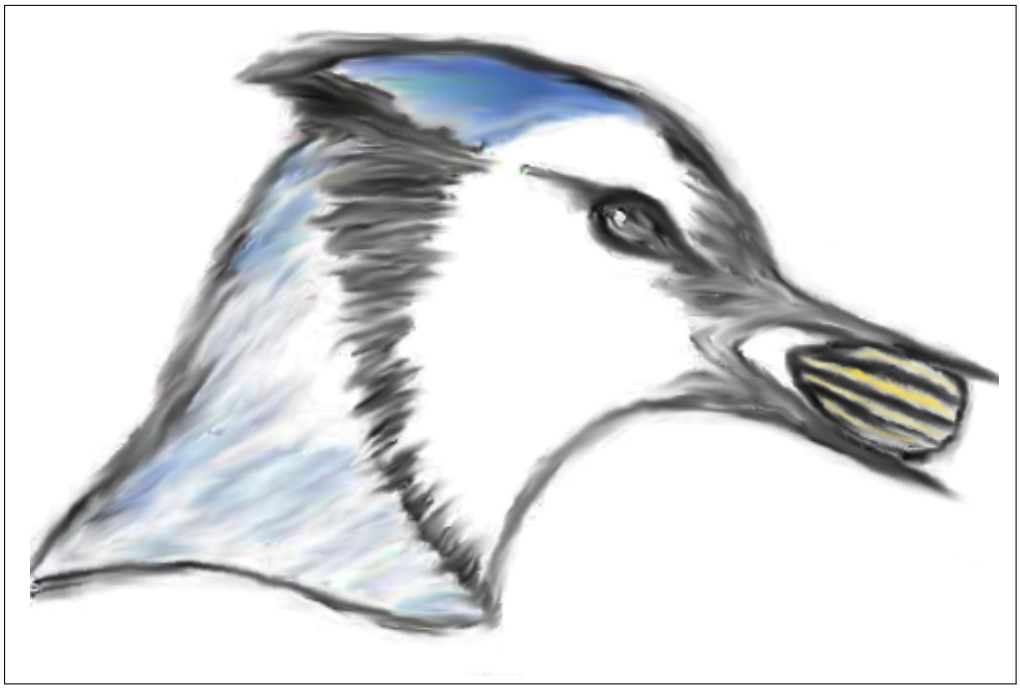
***** Lines 20-115 were ommitted in this abridged version. *****

```
      Subgroups have complete jurisdiction of the contents in their folder.
What is the sandbox folder for?
         Its for exploration purposes.
What folders in the sandbox am I allowed to use?
         The one with your name on it.

5. TRACKING WITH REDMINE
******************************
Where should I log my hours?
         On Redmine, go to the Log Hours link on the
         right side of the home page.

6. POLICY CHANGES
******************************
Is this document subject to change?
      Yes, team members can agree on adding, removing,
      or amending group policies in this document.
How does that work?
      A team member motions to add, remove or amend a
      policy. If the rest of the team consents to this
      decision, the policy is updated.
```

```
END File: /Team Documentation/Birdfeedr Policies.txt
```

BirdFeedr Logo

Description The BirdFeedr Logo is an image that represents our team. The image was first introduced at the beginning of the final project. It was liked enough by the team that we decided to use it as our logo. The following artifact is the image we decided upon to be our logo.



Exploration

Summary When we chose to work on an extension for BlueJ, we realized the team had no experience writing BlueJ extensions. We decided that we needed to learn about how they worked. Thus, we created two exploration projects: Project Git and Project Chirp. These projects helped us not only learn about BlueJ extensions, but also gain confidence at writing them.

Project Chirp

Description Project Chirp is a completed BlueJ extension that will cause BlueJ to play a chirping sound when a user compiles a file. Project Chirp was an exploration project that allowed the team to learn more about the BlueJ API and creating BlueJ extensions. During this project, the team learned several interesting quirks about BlueJ. For instance, we learned that BlueJ will store files in different locations depending on the OS. We also discovered that BlueJ provides CompileEvents and CompileListeners in their API. The following artifact consists of an exploration summary of what we learned while creating this extension.

```
BEGIN File: /Projects/Chirp/Exploration Summary.txt
```

```
Exploration Summary: Team Chirp
Jan. 29th, 2015

SUMMARY OF KNOWLEDGE LEARNED:
- BlueJ s API is composed of events and listeners: if a certain event happens,
  a listener can be implemented to find the event.
- The API has a CompileListener interface and a CompileEvent class.
- BlueJ s default path for finding files differs depending on the OS it is
installed on:
  - In Windows, it defaults to its install directory
  - In OSX, it defaults to the user s home directory
  - Use bluej.getSystemLibDir() and bluej.getUserConfigDir() to obtain
    the system s default library directory and user config directory
respectively.
  - To obtain an absolute path to the library s directory, use:
    bluej.getSystemLibDir().getAbsolutePath()

EXTENSION VERSIONS:
1. Implementation of SimpleExtension
2. Creates JBox upon successful compile
3. Creates JBox and plays designated .wav file upon successful compile
```

```
END File: /Projects/Chirp/Exploration Summary.txt
```

Project Git

Description Project Git is a semi-completed BlueJ extension that will allow the user to initialize a repository and commit. Project Git was created as an exploration project to allow the team to learn more about BlueJ extensions and its API. The following artifact consists of a sample of source code created for Project Git.

BEGIN File: /Projects/Git/Documents/User Stories.txt

```
1. As a user, I would like to save my work to a git repository on my computer.
TIME 1 WEEK
2. As a user, I would like to use Github with my BlueJ project. TIME 2 WEEKS
3. As a user, I would like to revert to a commit that I made earlier. TIME 3
WEEKS
4. As a user, I would like to create a new branch.  TIMES 2 WEEKS
5. As a user, I would like to checkout a repository using Git. TIME 1 DAYS but
it overlapps with other user stories
6. As a user, I would like to commit to a repository using Git. TIME 1 DAYS but
it overlapps with other user stories
7. As a user, I would like to use a Git repository in BlueJ with a GUI. TIME
0.2 WEEKS
8. As a user, I would like to save my work to a git repository on a remote
computer. OVERLAPPS WITH #2. Possible combine.
```

END File: /Projects/Git/Documents/User Stories.txt

BEGIN File: /Projects/Git/Git-Extension/README.TXT

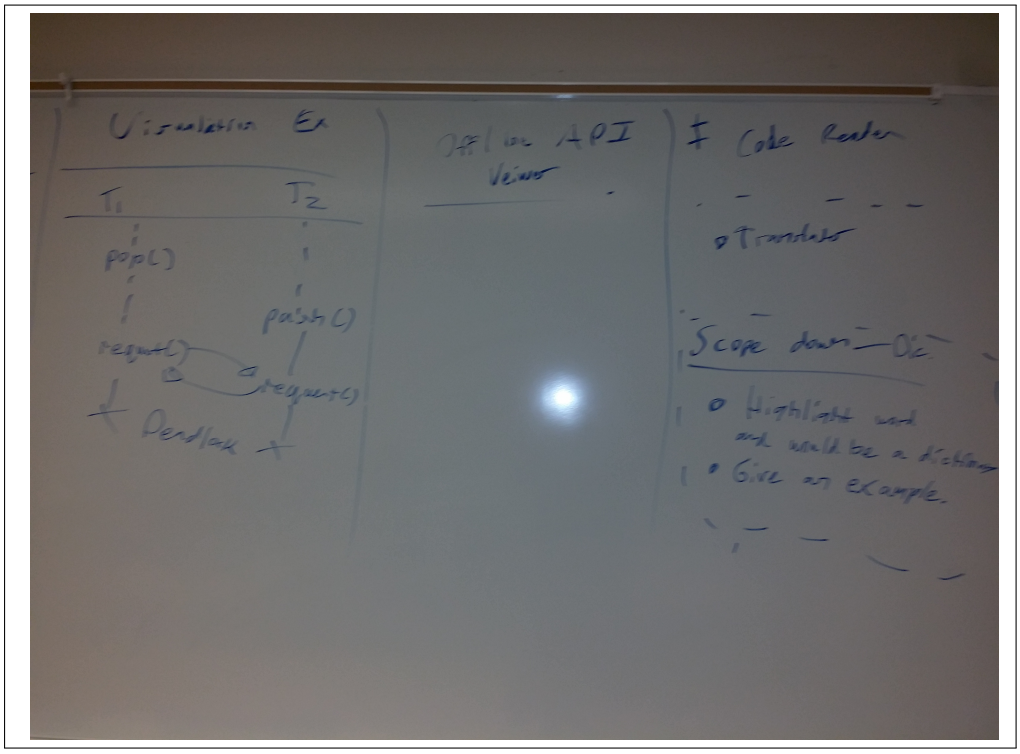
```
  BlueJ Git Extension
==========================================================================

Description
--------------------------------------------------------------------------
The BlueJ Git Extension allows BlueJ users to integrate
 with [Git](http://git-scm.com/).

Getting started
--------------------------------------------------------------------------
 . Create a Jar file and set the main class to "Git Extension" based
   in the manual section "Create (Executable) JAR Files"
   at http://www.bluej.org/doc/bluej-ref-manual.pdf

   Note: leave all options unchecked.

 . Install the extension based on the directions
    at http://www.bluej.org/extensions/extensions.html

Directory Structure
--------------------------------------------------------------------------
Git-Extension
            +lib            --- Jar dependencies.
         Actions            --- Menu Actions.
```

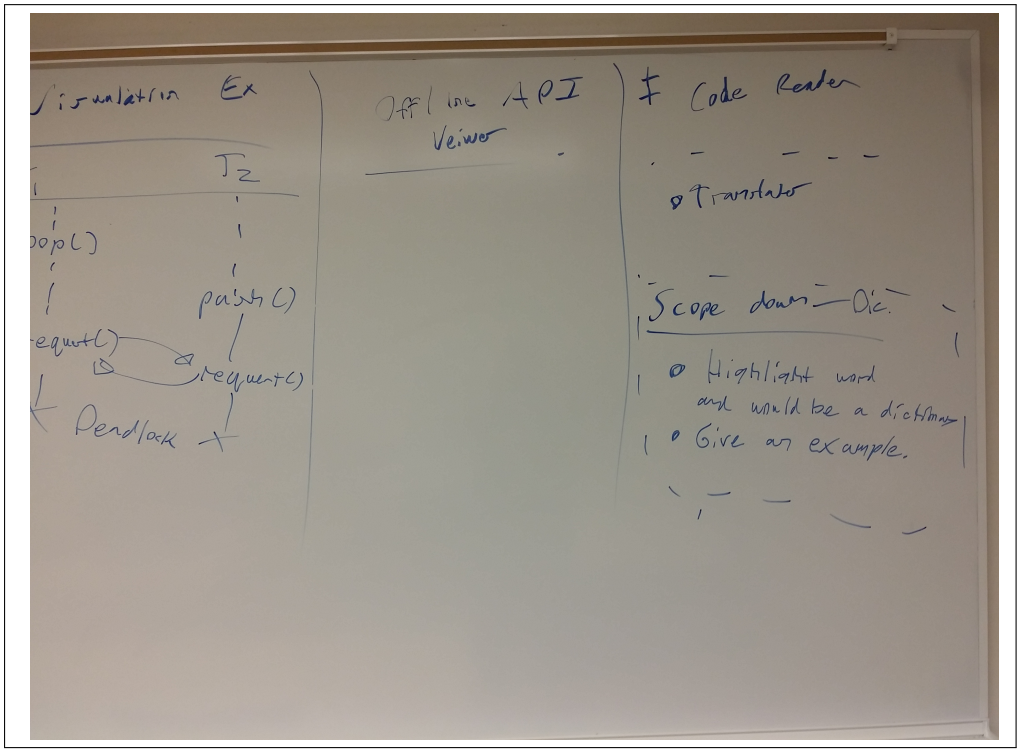
```
        Exceptions      --- Git Extension Exceptions.
         Helpers        --- Simplifies talking to BlueJ and JGit.

Authors
-------------------------------------------------------------------------------
 - Miguel Roman-Roman
 - Josh Gillham
 - Thomas Macari
```

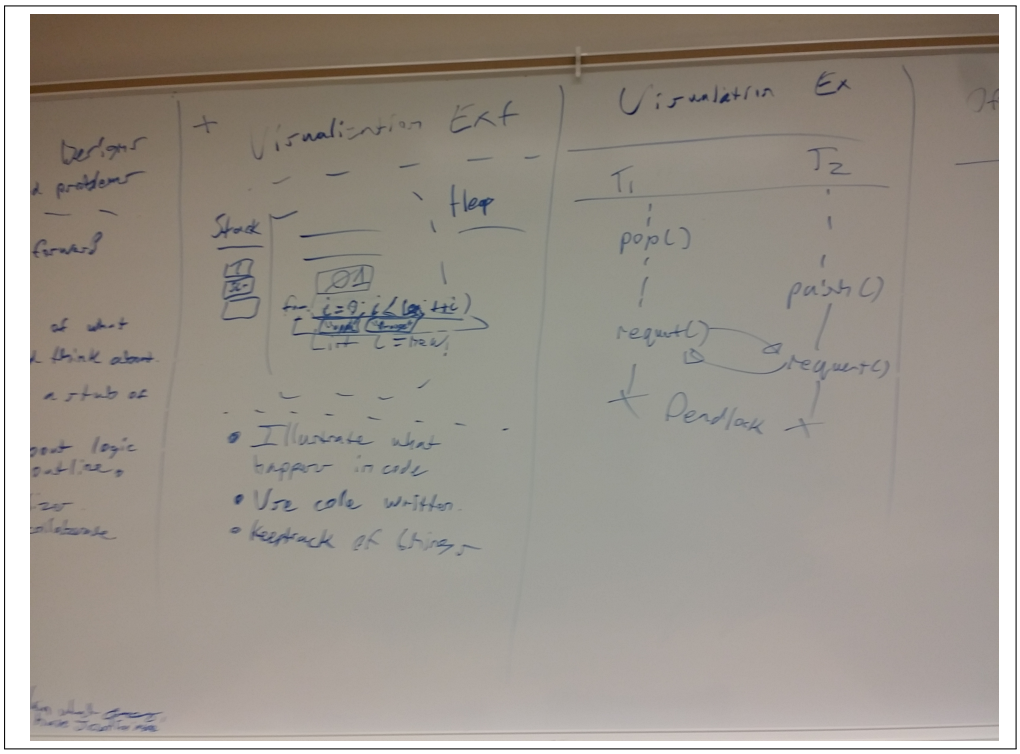
```
END File: /Projects/Git/Git-Extension/README.TXT
```

Surveys

Description This survey was used to inquire students about their programming experience, their biggest challenge, what helped them overcome their biggest challenge, and additional comments. The survey also includes thoughts from 2 professors about issues students usually face, what made it easier for the students, and ideas for possible project. We used the data from this survey to help guide us towards a final, informed decision for the main idea of our product. The following artifact consists of the results and notes of the survey.







Note: 7 artifacts were ommitted in this abridged version.

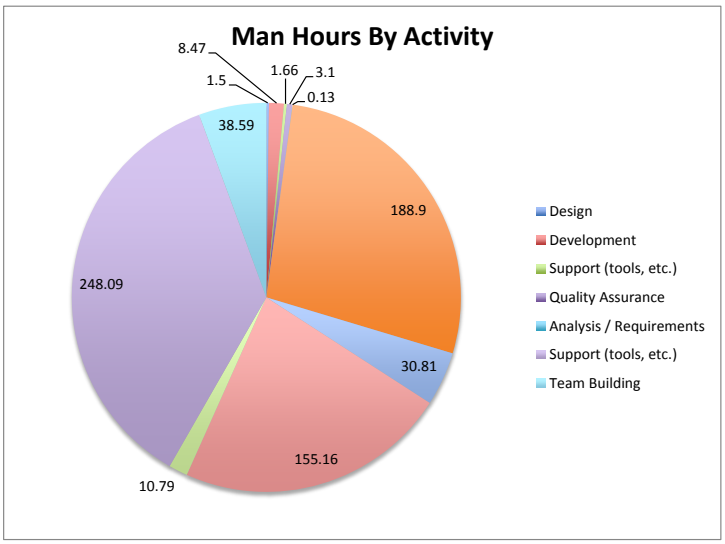
Metrics

Summary Team metrics came from a variety of sources like lines of code and logged time. This data was processed into reports for the purpose of presenting this information in our note book. Specifically for logged time, our group created metrics by logging the time that we spent by category. This data was recorded online in our Redmine project management tool.

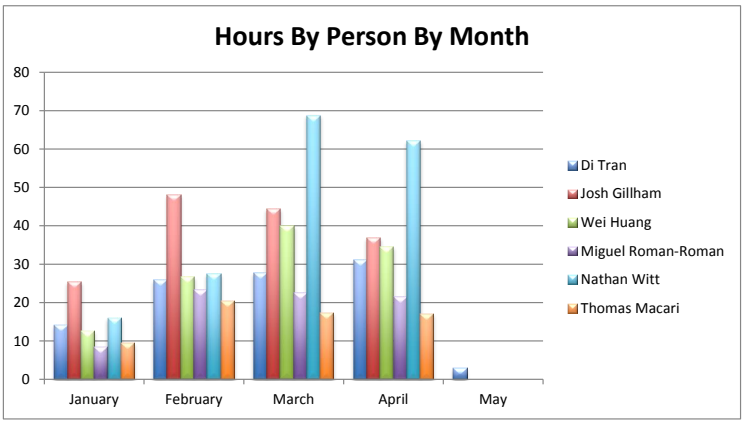
Timesheet

Description The following artifact consists of some statistics gathered regarding the teams total number of hours worked. The pie chart splits up our hours by activity, while the second chart splits up the hours by month and time.

Activity	Hours
Design	1.5
Development	8.47
Support (tools, etc.)	1.66
Quality Assurance	3.1
Analysis / Requirements	0.13
Analysis / Requirements	188.9
Design	30.81
Development	155.16
Quality Assurance	10.79
Support (tools, etc.)	248.09
Team Building	38.59
	687.21

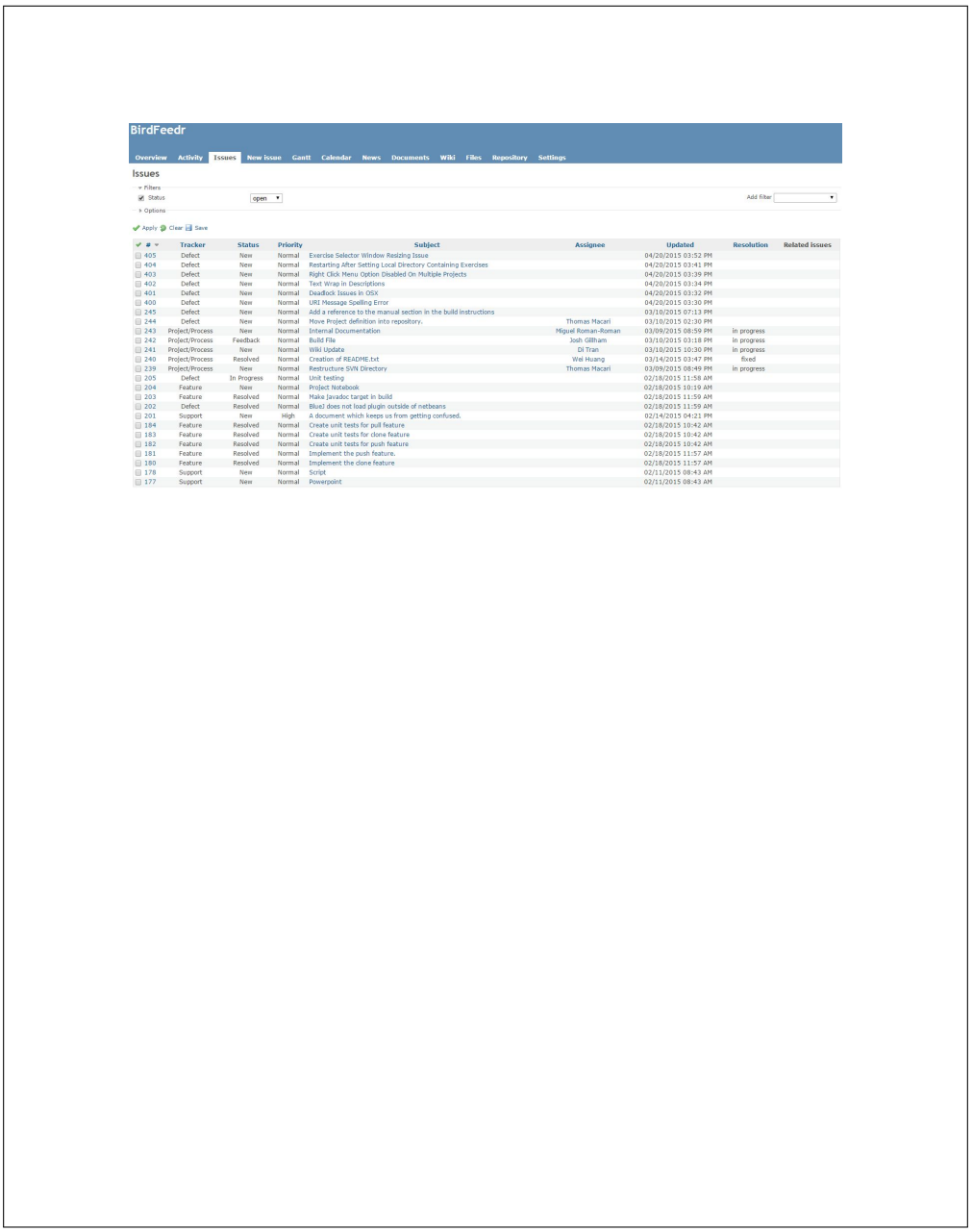


Name	January	February	March	April	May	Total time
Di Tran	14.25	25.91	27.7	31.15	3	102.01
Josh Gillham	25.38	48.14	44.36	36.93		154.8
Wei Huang	12.55	26.75	40	34.5		113.8
Miguel Roma	8.45	23.4	22.5	21.5		75.85
Nathan Witt	16	27.5	68.75	62		174.25
Thomas Maca	9.55	20.45	17.4	17.1		64.5
Total time	86.17	172.15	220.71	203.18	3	685.21



End-to-End Test

Description The following artifact is an image of our issues logged into our Redmine website. Many of the issues logged are from the results of an End-to-End test our team went through at the end of development of our BlueJ TA extension. We performed these tests by going through the entire extension from start to finish, and logged any issues found. These issues are used to keep track of known bugs our program suffers through. Since we do not have any more time to spare, we have these known bugs logged so any future developer knows what to expect to fix.



Size of product

Description The following artifact shows the number of classes within the BlueJ TA project as well as how many lines of code are in each class and the project as a whole. We created this artifact because we wanted another metric to show how much work was done on the project. The artifact is a text file containing the classes, the number of lines of codes in each class, and the total number of lines of code for the project.

```
BEGIN File: /Team Documentation/Reports/Report-BlueJ_TA-Lines-Of-Code.txt
```

```
  399 ../../Projects/BlueJ_TA/Code/src/Extension/BackEnd/Exercise.java
  151 ../../Projects/BlueJ_TA/Code/src/Extension/BackEnd/FileUtil.java
   74 ../../Projects/BlueJ_TA/Code/src/Extension/BackEnd/RemoveThread.java
   75 ../../Projects/BlueJ_TA/Code/src/Extension/BackEnd/runner/JRunner.java
  353 ../../Projects/BlueJ_TA/Code/src/Extension/BackEnd/StateManager.java
   76 ../../Projects/BlueJ_TA/Code/src/Extension/BackEnd/XMLReader.java
  127 ../../Projects/BlueJ_TA/Code/src/Extension/GUI/DescriptionGUI.java
   94 ../../Projects/BlueJ_TA/Code/src/Extension/GUI/ExerciseListGUI.java
   91
../../Projects/BlueJ_TA/Code/src/Extension/GUI/FXMLDescriptionDocumentController
.java
  110
../../Projects/BlueJ_TA/Code/src/Extension/GUI/FXMLExerciseDocumentController.ja
va
  100
../../Projects/BlueJ_TA/Code/src/Extension/GUI/FXMLTestResultsDocumentController
.java
  128 ../../Projects/BlueJ_TA/Code/src/Extension/GUI/TestResultsGUI.java
   77 ../../Projects/BlueJ_TA/Code/src/Extension/Main.java
   94 ../../Projects/BlueJ_TA/Code/src/Extension/MenuBuilder.java
  157 ../../Projects/BlueJ_TA/Code/src/Extension/Preferences.java
 2106 total
```

```
END File: /Team Documentation/Reports/Report-BlueJ_TA-Lines-Of-Code.txt
```

The size of the project in lines of code

Description The report below shows the size of project in lines of code. It shows each files size and the total size of the extension.

```
BEGIN File: /Team Documentation/Reports/Report-Commits.txt
```

```
------------------------------------------------------------------------
r300 | jgillham | 2015-05-01 17:12:52 -0600 (Fri, 01 May 2015) | 1 line

Updated the READMEs and styled.
------------------------------------------------------------------------
r299 | jgillham | 2015-05-01 16:45:29 -0600 (Fri, 01 May 2015) | 1 line

Removed unprinted funny chars in log.
------------------------------------------------------------------------
r298 | jgillham | 2015-05-01 16:44:42 -0600 (Fri, 01 May 2015) | 1 line

Added new READMEs for Project BlueJ TA.
------------------------------------------------------------------------
r297 | jgillham | 2015-05-01 16:38:11 -0600 (Fri, 01 May 2015) | 1 line

Renamed root readme to README.
------------------------------------------------------------------------
r296 | jgillham | 2015-05-01 09:58:11 -0600 (Fri, 01 May 2015) | 1 line

Wrote script to grab updates from Google drive, but, should be used with
```

***** Lines 20-1310 were ommitted in this abridged version. *****

```
------------------------------------------------------------------------
r5 | mroman | 2015-01-29 13:01:40 -0700 (Thu, 29 Jan 2015) | 1 line

Test of basic extension
------------------------------------------------------------------------
r4 | jgillham | 2015-01-28 20:36:29 -0700 (Wed, 28 Jan 2015) | 1 line

Updated the build with today's work.
------------------------------------------------------------------------
r3 | jgillham | 2015-01-28 20:33:44 -0700 (Wed, 28 Jan 2015) | 1 line

Added file from the basic extension.
------------------------------------------------------------------------
r2 | jgillham | 2015-01-28 19:44:52 -0700 (Wed, 28 Jan 2015) | 1 line

Uploaded meeting notes.
------------------------------------------------------------------------
r1 | tmacari | 2015-01-28 09:40:51 -0700 (Wed, 28 Jan 2015) | 1 line

Created the initial base directories.
------------------------------------------------------------------------
```

```
END File: /Team Documentation/Reports/Report-Commits.txt
```

Process

Summary Our group went through an iterative and incremental process and we tried to follow what we learned about process models from the previous course. Our actual process borrowed characteristics from various process models. For examples, we used XP's concept of user stories and throw away models. For another example, we used the Spiral's model concept of prototyping to manage design risks.

Vision Statement Heuristic

Description The vision statement heuristic is an adaptation to the naming heuristic. This adaptation was used to produce our current vision statement. The following artifact consists of the notes and method used when creating the team vision statement.

BEGIN File: /Team Documentation/Team Portfolio/Artifacts/Process/Vision Statement Heuristic.txt

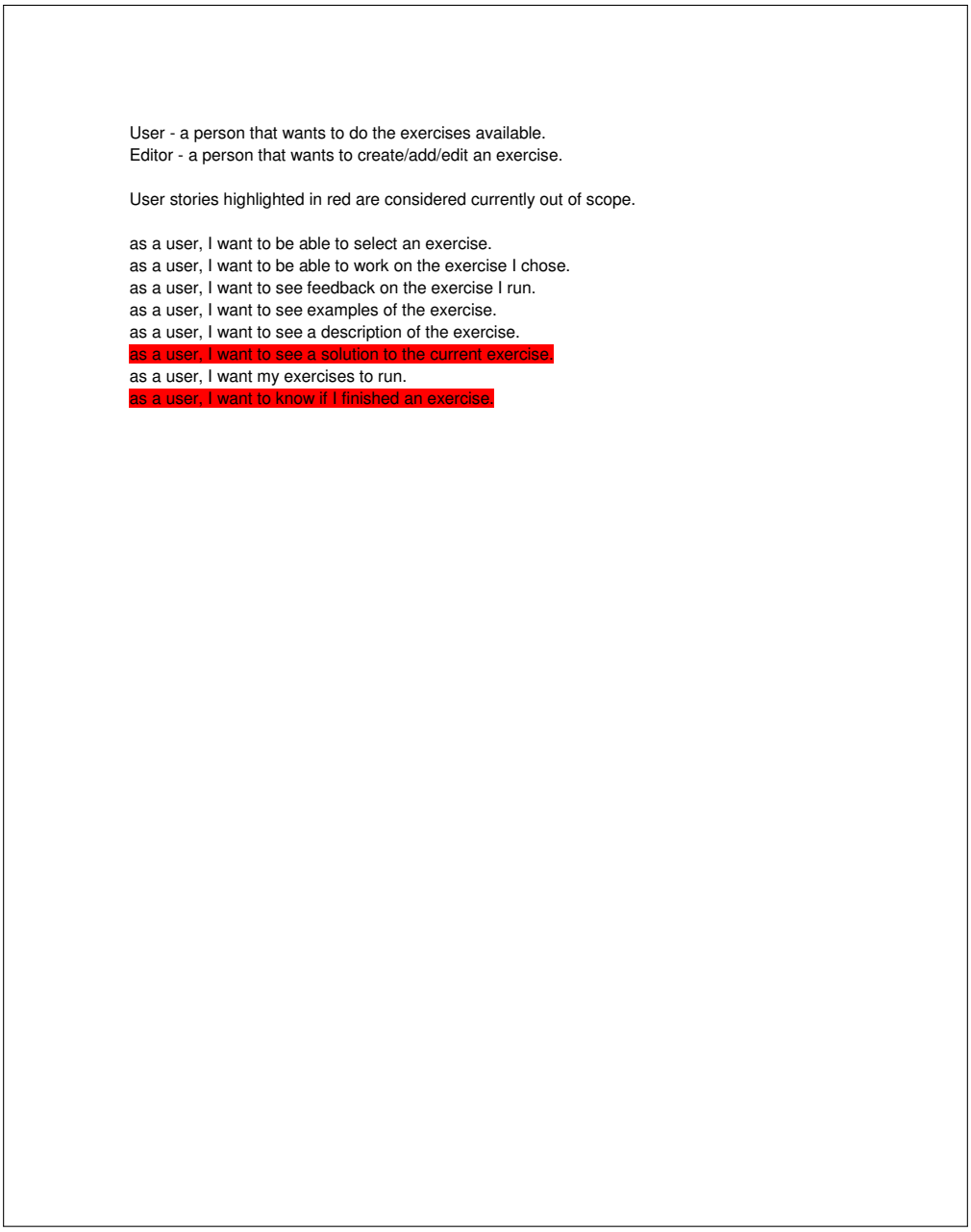
---

How We Came Up With the Vision Statement
Some of you may be wondering how we came up with the vision statement. We used the naming heuristic presented from The Project s Name.
What is the naming heuristic? Here s how you do it:
Come up with a name
Think of three reasons why the name won t work.
Create a new name that addresses these three things.
Repeat this process until you come up with a satisfying name.1

END File: /Team Documentation/Team Portfolio/Artifacts/Process/Vision Statement Heuristic.txt

User Stories

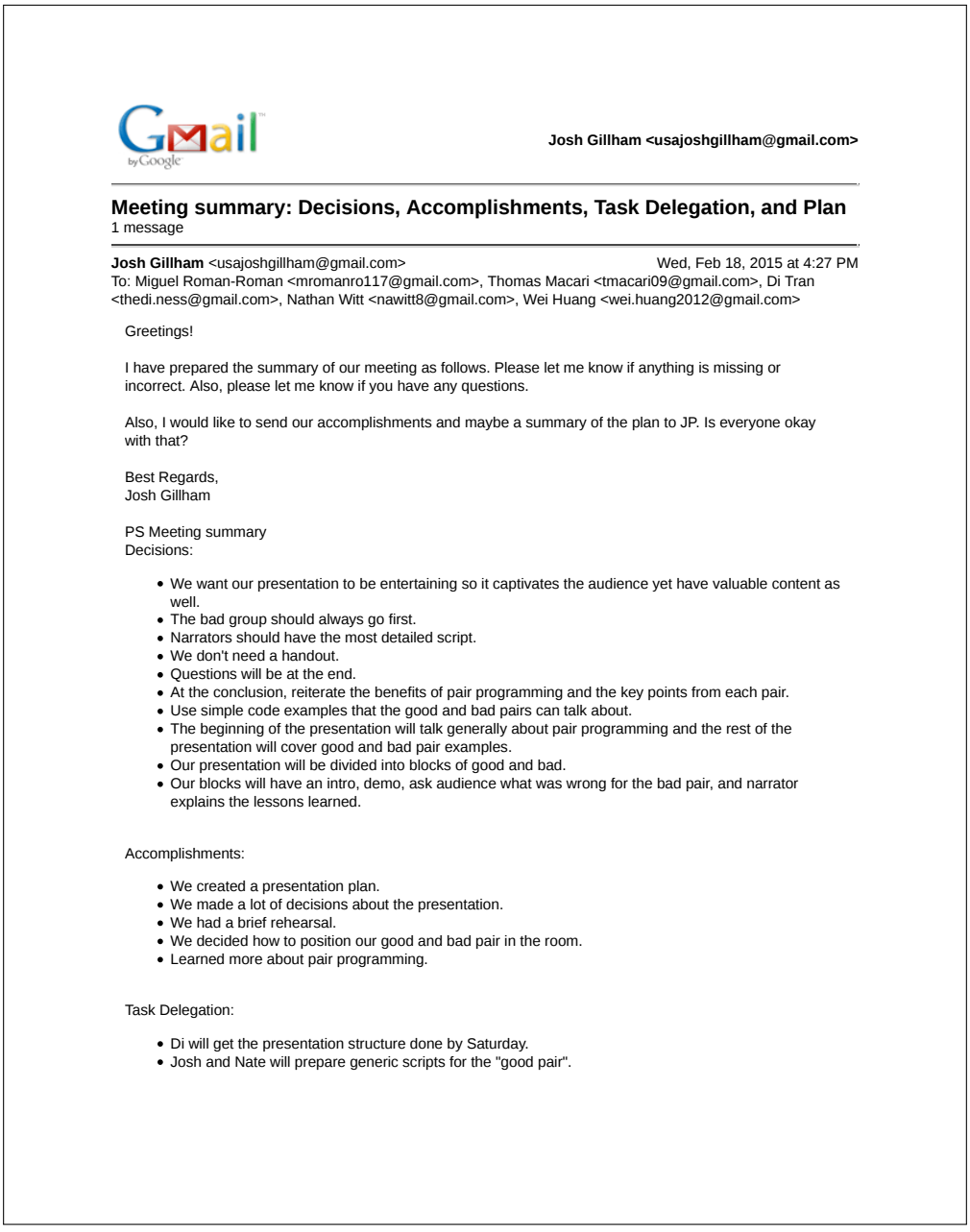
Description This artifact is a set of user stories that the team came up with. We created user stories to serve as requirements for the project. The user stories helped the team scope down the extensions functionalities to a concrete list of scenarios, and overall these stories allowed the team to share a common goal, even when the team split up.



Meeting Summaries

Description The meeting summaries usually consisted of an email sent to all members after an important meeting to help perception alignment as well as for documentation of our product and process. Our summaries typically included information like comments from our professor, decisions, accomplishments, and delegated tasks.

The following artifacts consists a archive of our meetings.



Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Meeting summary: Decisions, Accomplishments, Task Delegation, and Plan
1 message

Josh Gillham <usajoshgillham@gmail.com> Wed, Feb 18, 2015 at 4:27 PM
To: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Greetings!

I have prepared the summary of our meeting as follows. Please let me know if anything is missing or incorrect. Also, please let me know if you have any questions.

Also, I would like to send our accomplishments and maybe a summary of the plan to JP. Is everyone okay with that?

Best Regards,
Josh Gillham

PS Meeting summary
Decisions:

- We want our presentation to be entertaining so it captivates the audience yet have valuable content as well.
- The bad group should always go first.
- Narrators should have the most detailed script.
- We don't need a handout.
- Questions will be at the end.
- At the conclusion, reiterate the benefits of pair programming and the key points from each pair.
- Use simple code examples that the good and bad pairs can talk about.
- The beginning of the presentation will talk generally about pair programming and the rest of the presentation will cover good and bad pair examples.
- Our presentation will be divided into blocks of good and bad.
- Our blocks will have an intro, demo, ask audience what was wrong for the bad pair, and narrator explains the lessons learned.

Accomplishments:

- We created a presentation plan.
- We made a lot of decisions about the presentation.
- We had a brief rehearsal.
- We decided how to position our good and bad pair in the room.
- Learned more about pair programming.

Task Delegation:

- Di will get the presentation structure done by Saturday.
- Josh and Nate will prepare generic scripts for the "good pair".

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

In class summary: Accomplishments, Plan, Assignments, Decisions

1 message

Josh Gillham <usajoshgillham@gmail.com> Mon, Feb 23, 2015 at 5:09 PM
To: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Greetings!

I have included a post script summary of our in-class work today. Please let me know if I missed anything or if you have any questions.

Best Regards,
Josh Gillham

PS Meeting Summary

Accomplishments

- Filtered inappropriate projects.
- Rehearsed twice.
- Planned Tuesday and Wednesday.
- Presentation was timed at 10 min 45 secs not including audience response.

Plan

- Tuesday
  - Presentation artifacts will have been polished and proofread.
- Wednesday
  - Rehearse 30 min before class.
  - Presentation.
  - Reflect on our presentation feedback.
  - Discuss our next presentation.

Assignments

- (Optional) Each person thinks of an additional project.
- Each person picks their favorite project.
- Each person writes a very short (< 150 word) personal informal report to support their favorite.
- Nate will work on the cast slide and animations.
- Di, Wei, Josh are doing Quality Assurance.
- Thomas is working on the transitions.
- Miguel is polishing the narrator script.

Decisions

- Stage Props
  - Good pair uses Nate's PC.
  - Handicap desk.
  - Bad pair is bring both PC's

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

In Class Meeting Summary: Informal Consensus on project ideas, Plan, and more

6 messages

Josh Gillham <usajoshgillham@gmail.com> Wed, Feb 25, 2015 at 4:52 PM
To: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Greetings!

I have included a meeting summary post script. Di and Nate I was thinking that this would be last internal email you guys receive until we rejoin if that is okay with you both. To everyone, please respond with your comments and questions.

Di and Nate, I forgot to ask you about how working together on a presentation in the future would work. Tell me what you think.

Best Regards,
Josh Gillham

PS Meeting summary

Informal Consensus on project ideas:

- Everybody likes interpretter, but, JP said its already in BlueJ so we got shut down.
- Everybody likes the stack and thread visualization ideas.

Plan

1. Find out what stack holders want. Thus, we will conduct surveys to identify the problem students are having.
2. Once we identify problems, we can see which project ideas will solve the problems (if any).
3. Brainstorm new projects as needed.
4. We will pick one project and research the feasibility. This will make sure we can do the project.
5. If we can't do the project then we will back up and reassess the scope and possible repeat #4.
6. If we can do the project then we will pick a process model and work on the first step of that process model.

Presentation Feedback

- Make the subtitle more profound i.e. How to get the most out of pair programming.
- Last slide with the cast was not such a great ending. Instead end with the messages.
- Audience might want to see something on the screen about what the pair was working on.
- Begin with the conclusion.
- Include citations.
- Focus on core message.
- Keep the slides in sync.
- Give a larger picture on the topic of the presentation.

Note: 7 artifacts were ommitted in this abridged version.

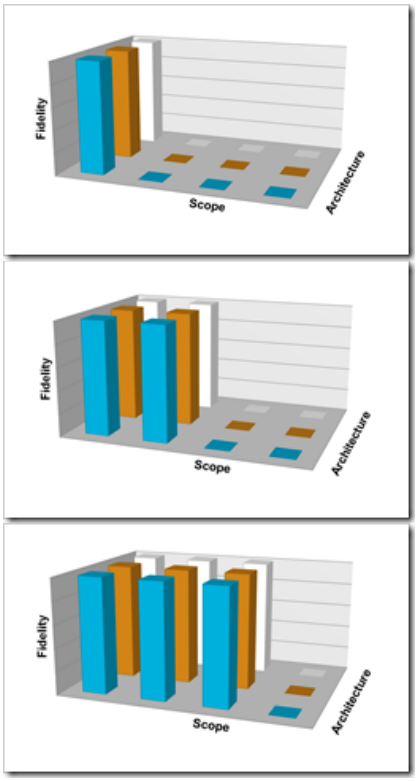
Software Development Life Cycle - Iterative and Incremental Model

Description Our process model was informal, yet, it borrowed characteristics from popular process models such as XP, and the Spiral models. From XP, we borrowed the concepts of user stories, release planning, and throw away design metaphors. From the Spiral model, we borrowed the concept of prototypes. The purpose of the model was to think about real process models and how we related to them. It was also to help us gain direction on how we would create our project. The following artifact shows a graph of how our process model looked like in terms of fidelity, scope, and architecture.

Incremental

The purely incremental approach builds each feature, across all components, to full fidelity, one by one.

(http://availagility.co.uk/wp-content/uploads/2009/12/Incremental1.png)

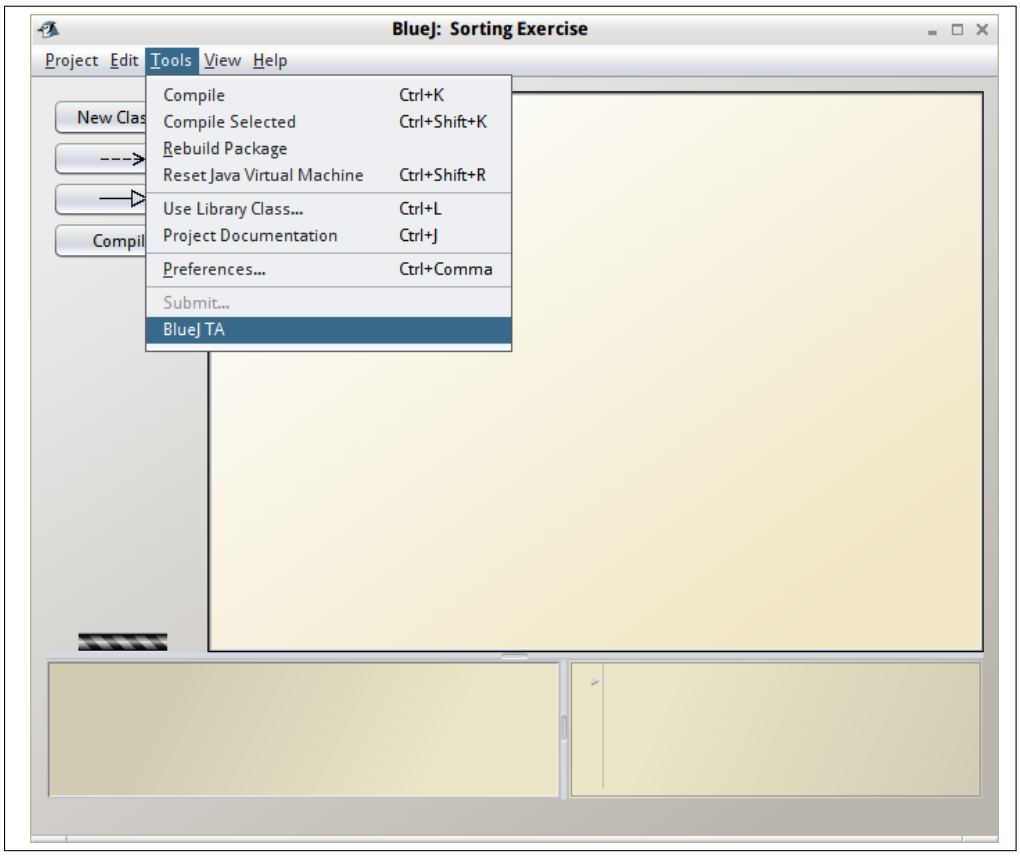


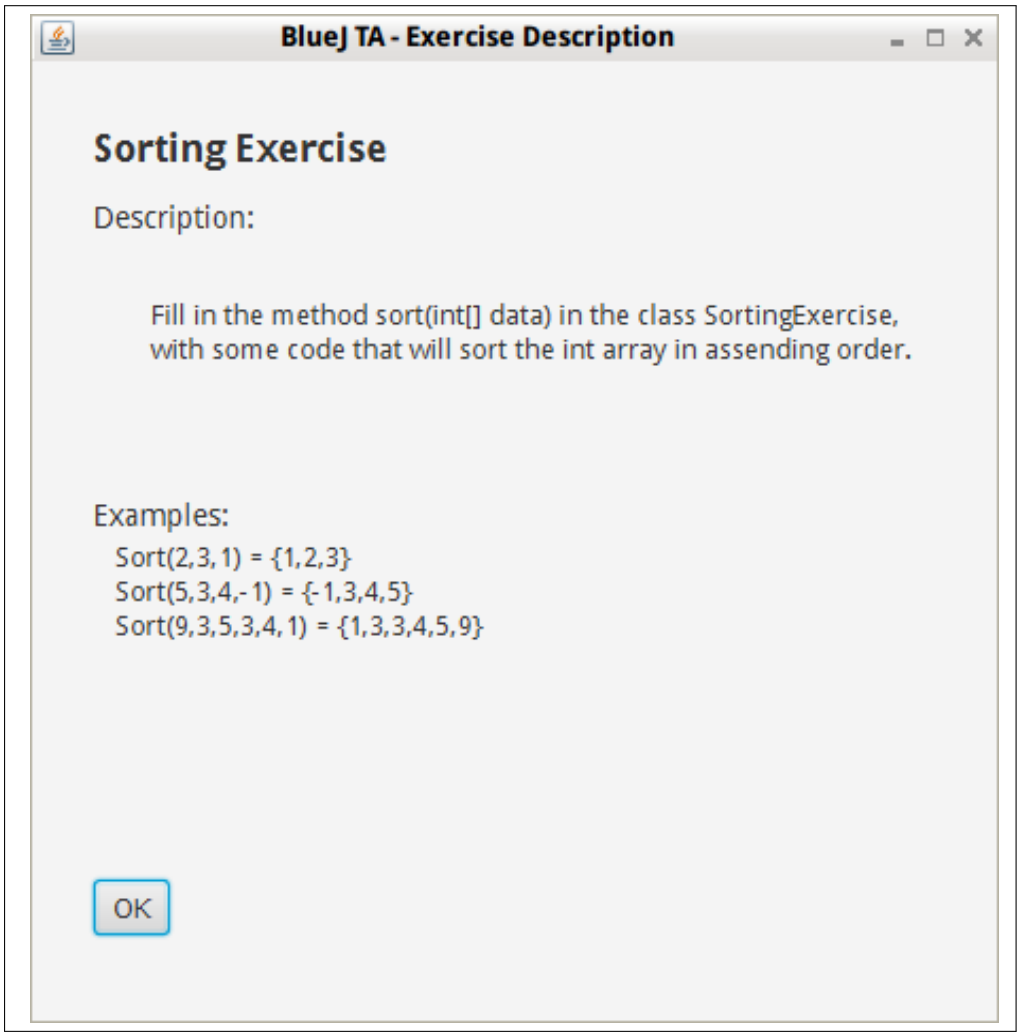
Product

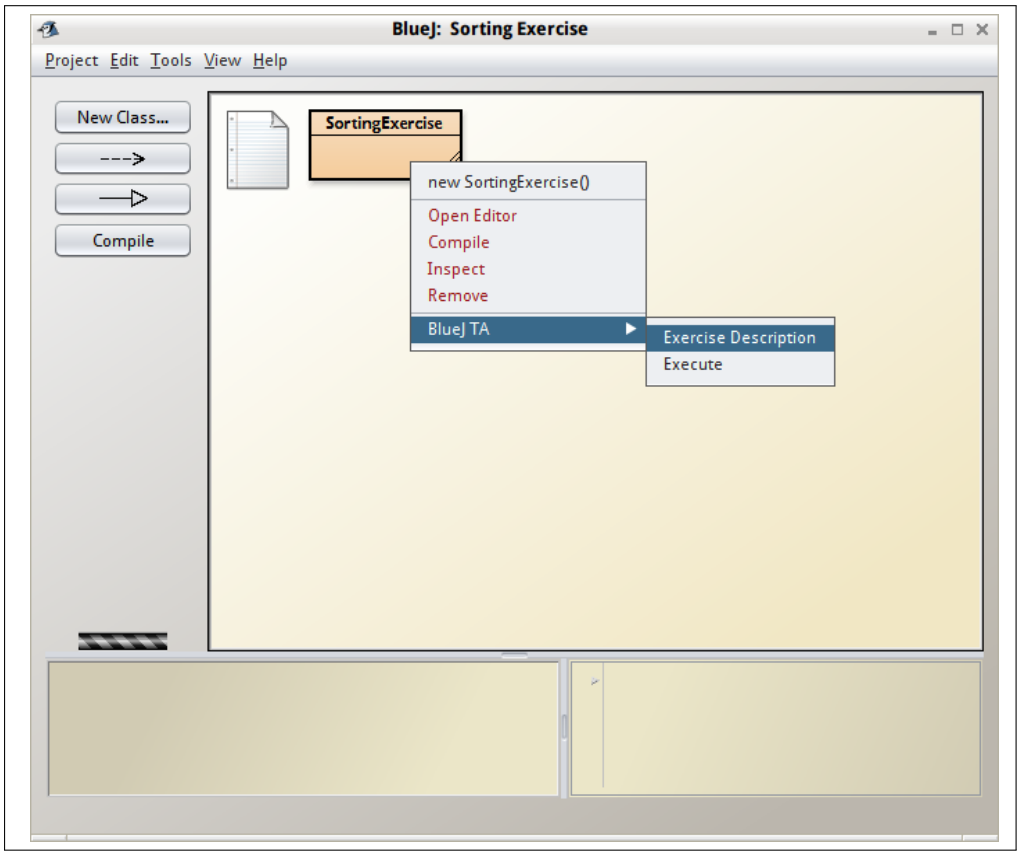
Summary Our product, BlueJ TA, is a BlueJ extension that aids in the learning process by providing programming exercises for students to complete and verify in real-time. The following section describes the elements that as a whole compose our product. Other than the BlueJ extension menus, our project was split into three major modules: the JUnit test runner, the .xml reader, and the set of graphical user interfaces. We also created some side auxiliary artifacts to aid the main project, such as sample exercise files to be shipped with the product and a build file to help with the cleaning and jaring of the product.

BlueJ TA

Description The following artifact is the BlueJ TA extension our team developed. This product is composed of three main modules: the graphical user interfaces, the XML parser, and the JUnitRunner. This program was the outcome of a 16 week project for the CS 4260 course. This program is in a fully functional state, though there are some parts that were not fully implemented due to the short amount of time we had to complete the project.




BlueJ TA - Exercise Description
Sorting Exercise
Description:
Fill in the method sort(int[] data) in the class SortingExercise,
with some code that will sort the int array in assending order.
Examples:
Sort(2,3,1) = {1,2,3}
Sort(5,3,4,-1) = {-1,3,4,5}
Sort(9,3,5,3,4,1) = {1,3,3,4,5,9}
OK



Note: 5 artifacts were ommitted in this abridged version.

The Java sources for Project BlueJ TA

Description The sources files are the actual program code of the extension. We have included a small sample below.

BEGIN File: /Projects/BlueJ_TA/Code/src/Extension/BackEnd/Exercise.java

---

```
package Extension.BackEnd;

import bluej.extensions.*;
import java.awt.Frame;
import java.util.List;
import javax.xml.bind.annotation.*;

import java.io.*;

/**
 * An Exercise object holds all the needed information related to the exercise.
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
```

***** Lines 20-379 were ommitted in this abridged version. *****

```
        for (int x = 0; x < example.size(); x++) {
            System.out.println("Example " + x + ": " + example.get(x));
        }
        System.out.println("Java Name: " + javaName);
        System.out.println("Java Code: " + javaCode);
        System.out.println("Junit Name: " + junitName);
        System.out.println("Junit Code: " + junitCode);
    }

    /**
     * Used primarily to populate the Observable list in the Exercise List GUI with the titles of the
     * exercises
     *
     * @return  this exercise's title
     */
    @Override
    public String toString()
    {
        return this.title;
    }
}
```

END File: /Projects/BlueJ_TA/Code/src/Extension/BackEnd/Exercise.java

BEGIN File: /Projects/BlueJ_TA/Code/src/Extension/BackEnd/FileUtil.java

---

```
package Extension.BackEnd;

import bluej.extensions.*;
import java.io.*;

/**
 * Utility methods for dealing with files and BlueJ projects.
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public abstract class FileUtil
{
    /**
     * Saves the given contents into the given file specified by path.
```

***** Lines 20-131 were ommitted in this abridged version. *****

```
     *
     * @param bluej the running bluej program.
     * @param name the name of the project to close.
     */
    public static void closeProj(BlueJ bluej, String name)
    {
        for (BProject proj : bluej.getOpenProjects())
        {
            try
            {
                if (proj.getName().equals(name))
                {
                    proj.close();
                }
            } catch (Exception e)
            {
                System.out.println("Error: " + e);
            }
        }
    }
}
```

END File: /Projects/BlueJ_TA/Code/src/Extension/BackEnd/FileUtil.java

BEGIN File: /Projects/BlueJ_TA/Code/src/Extension/BackEnd/RemoveThread.java

---

```
package Extension.BackEnd;

import java.awt.Frame;
import bluej.extensions.BClass;
import bluej.extensions.BField;
import bluej.extensions.BObject;

/**
 * A thread used to clean up execution artifacts within a users project and unlock
 * its ui once they are loaded into memory.
 *
 * @author Team BirdFeedr
 */
public class RemoveThread extends Thread
 {

    //the frame to unlock after removal of the classes
    private final Frame frame;

    //the classes to remove
```

***** Lines 20-54 were ommitted in this abridged version. *****

```
            System.out.println("Err: " + e);
        }
        try
        {
            testClass.remove();
        }
        catch (Exception e)
        {
            System.out.println("Error: " + e);
        }
        try
        {
            runClass.remove();
        }
        catch (Exception e)
        {
            System.out.println("Error: " + e);
        }
        frame.getComponent(0).setVisible(true);
    }
}
```

```
END File: /Projects/BlueJ_TA/Code/src/Extension/BackEnd/RemoveThread.java
```

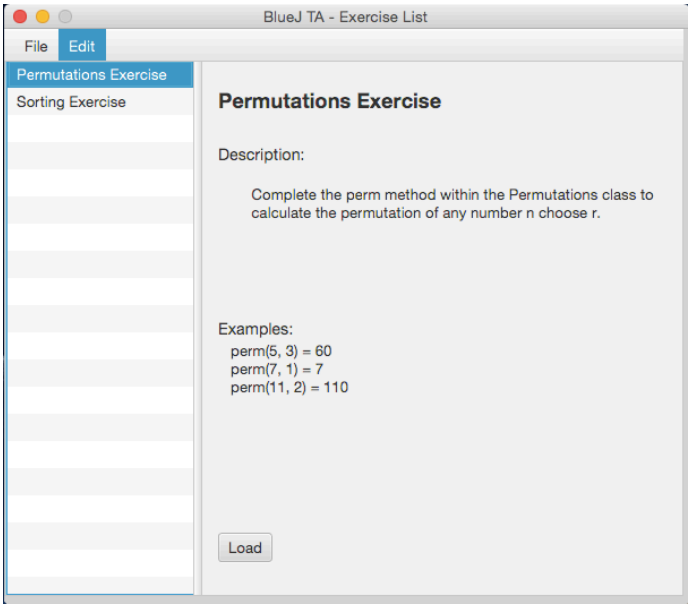
Note: 12 artifacts were ommitted in this abridged version.

Graphical User Interface for BlueJ TA

Description BlueJ TA used a total of three different graphical user interfaces coded using JavaFX. Where the first was used to allow a user to pick their exercise, the second was used to display descriptive info about an exercise to the user, and the third was used to deliver test results to the user after execution of an exercise.

We decided to use JavaFx to create these GUI's because of the ease of creation with the tools available as well as the ease of maintained of the code. We arrived at three distinct GUI's with one feature per GUI because it seemed in-line with BlueJ's design nature.

The following artifacts consists of three images showing the three different windows used within BlueJ TA.



XML Parser for BlueJ TA

Description The following artifact is a sample of the code for the XML parser module created for the BlueJ TA extension. The XML parser module reads XML documents and loads the contents into memory. To be more specific, the XML parser reads in pre-generated XML exercise files and created Exercise objects for use by the extension. The parser uses the Java Jaxby tool to parser XML documents.

BEGIN File: /Projects/BlueJ_TA/Code/src/Extension/BackEnd/XMLReader.java

---

```
package Extension.BackEnd;

import java.io.File;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;

import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.FileInputStream;

/**
 * XMLReader class is used to read in XML files and convert them into Exercise objects.
 * This class assumes the XML file passed in is a BlueJ TA exercise.
 * Uses JAXB
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
```

***** Lines 20-56 were ommitted in this abridged version. *****

```
        return ex;
    }

    /**
     * Reads in an XML file from the given path
     *
     * @param filePath  the location of the exercise
     * @return  the Exercise object created from the given XML file.
     */
    public Exercise readExercise(String filePath)  {
       try
       {
           return readExercise( new FileInputStream( filePath ) );
       } catch ( Exception  e )
       {
           System.out.println("Error: " + e);
           e.printStackTrace();
           return null;
       }
    }
}
```

END File: /Projects/BlueJ_TA/Code/src/Extension/BackEnd/XMLReader.java

JUnitRunner for BlueJ TA

Description The JUnit Runner is a subset of code in BlueJ TA that runs a compiled JUnit test with a variety of dynamically created user classes. This is done by using the JUnitCore run feature from the JUnit API within BlueJ's user VM. Along with other elements of the project this aspect of BlueJ TA evolved over many different iterations to fix bugs and accommodate changes in requirements.

The following artifacts consists of the code that encompasses JUnit Runner.

BEGIN File: /Projects/BlueJ_TA/Code/src/Extension/BackEnd/runner/JRunner.java

---

```
/**
 * Note) This class cannot have a package header when it is compiled to work,
 * this is because this class is deployed to a default package to be run.
 *
 * This header was left in for anyone editing the file within BlueJ for BlueJ to
 * see the Java File. If left in this header will need to be commented out at
 * compile time either manually or within a build.
 */
//package Extension.BackEnd.runner;

import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

/**
 * A mobile class to be deployed into a users exercise project and called by the
 * extension to execute a JUnit test within the same project.
 *
 * @author Team BirdFeedr
 */
```

***** Lines 20-55 were ommitted in this abridged version. *****

```
        //Set the running flag here to let the extension know that both
        //the JRunner and JUnitTest are loaded into memory and its ok for
        //the underlying class files to be removed from the exercise project.
        running = true;

        Result result = new JUnitCore().run(testC);
        if (result.getFailureCount() == 0)
        {
            return "All test Passed!";
        } else
        {
            String msg = "";
            for (Failure fail : result.getFailures())
            {
                msg += fail.getMessage() + "\n";
            }
            return msg;
        }
    }
}
```

END File: /Projects/BlueJ_TA/Code/src/Extension/BackEnd/runner/JRunner.java

ANT Build File

Description The following artifact is the ANT build file created to build the BlueJ TA extension. This ANT build file provides recipe that ANT will use to build the extension. This build file works for the Windows, Mac OS X, and Linux operating systems. This build file was created so that the program could be run in a more efficient manner.

BEGIN File: /Projects/BlueJ_TA/Code/build.xml

```
<!--
This file is a build recipe for the project. It will compile Java files and
 install the extension into the user's home folder. Before building the project,
 please ensure that Apache Ant and the JDK are installed. This project has been
 tested with Ant version 1.9.3. It may work with older/newer versions, but, we
 can't make any guarantees. To build the project, run the "ant" command in the
 project folder. In Windows, open the command line and on Mac/Linux open the
 terminal.

Note:
Please use the "ant clean" command should be run to eradicate the old files.
 Developers should remember to run this command the first time or every time
 they update their SVN folder.

Also, as a developer, make sure that pre-existing extension JARs are deleting.
 Failure to remove pre-existing files may result in a false positive for
 building and running the extension.

Author: Josh Gillham
Version: 2015-04-18
```

***** Lines 20-272 were ommitted in this abridged version. *****

```
        <echo message = "END BlueJ's output."/>
        <echo message = "-----------------------------------------------"/>
        <waitfor>
            <available file="${bluej.log.file}"/>
        </waitfor>
        <copy file = "${bluej.log.file}" todir = "${results.dir}" />
    </target>

    <!-- Prints and saves BlueJ's output. -->
    <target name = "run" depends = "capture-log"
     description = "Performs all necessary actions to launch BlueJ then shows
the debug log."/>

    <!-- Generates Java Docs -->
    <target name = "doc" depends = "compile"
        description = "Generates Java documentation.">
        <javadoc sourcepath = "${source.dir}" destdir = "${doc.dir}">
            <classpath refid = "cp"/>
        </javadoc>
    </target>
</project>
```

END File: /Projects/BlueJ_TA/Code/build.xml

BlueJ TA Exercises

Description The following artifacts consist of two XML exercise files created for the use of BlueJ TA. These artifacts were created to test various functionalities of the BlueJ TA extension as well as to give reference as to how exercises must be written.

BEGIN File:
/Projects/BlueJ_TA/Code/src/Extension/exercises/PermutationExercise.xml

---

```
<?xml version="1.0" encoding="UTF-8"?>
<exercise>
    <title>Permutations Exercise</title>
    <description>
        Complete the perm method within the Permutations class to
        calculate the permutation of any number n choose r.
    </description>
    <examples>
        <example>perm(5, 3) = 60</example>
        <example>perm(7, 1) = 7</example>
        <example>perm(11, 2) = 110</example>
    </examples>
    <hint>
        Recursion might be a good approach to solving this problem in a simple
way.
    </hint>
    <sampleAnswer>
        public int perm(int n, int r) {
            if (r > 0) {
                return n * perm(n - 1, r - 1);
```

***** Lines 20-52 were ommitted in this abridged version. *****

```
            public void permutationTestC(){
                assertPerm(7, 1);
            }

            private static void assertPerm(int n, int r){
                int up = userPerm.perm(n, r);
                int tp = perm(n, r);
                assertTrue(&quot;perm(&quot;+ n +&quot;, &quot;+ r +&quot;) was
suppost to equal &quot;+ tp +&quot;, but got &quot;+ up +&quot; instead.&quot;,
up == tp);
            }

            private static int perm(int n, int r) {
                if (r > 0) {
                    return n * perm(n - 1, r - 1);
                }
                return 1;
            }
        }
    </junitCode>
</exercise>
```

END File:
/Projects/BlueJ_TA/Code/src/Extension/exercises/PermutationExercise.xml

BEGIN File: /Projects/BlueJ_TA/Code/src/Extension/exercises/SortExercise.xml

---

```
<?xml version="1.0" encoding="UTF-8"?>
<exercise>
    <title>Sorting Exercise</title>
    <description>
        Fill in the method sort(int[] data) in the class SortingExercise,
        with some code that will sort the int array in assending order.
    </description>
    <examples>
        <example>Sort([2, 3, 1]) -> [1, 2, 3]</example>
        <example>Sort([5, 3, 4, -1]) -> [-1, 3, 4, 5]</example>
        <example>Sort([9, 3, 5, 3, 4, 1]) -> [1, 3, 3, 4, 5, 9]</example>
    </examples>
    <hint>
        Think about how the data can be compared and switched between the other
elements.
    </hint>
    <sampleAnswer>
        public Class sort{
            public static void sort(int[] data){
                for(int i = 0; i &lt; data.length; i++){
```

***** Lines 20-67 were ommitted in this abridged version. *****

```
            }

            private static void assertSorted(int[] data, boolean assending) {
                for(int i = 0; i &lt; data.length - 2; i++){
                    boolean bool = (assending &amp;&amp; data[i] &lt;= data[i +
1]) ||
                                    (!assending &amp;&amp; data[i] &gt;= data[i
+ 1]);
                    if(!bool){
                    String comp = assending ? &quot;&lt;=&quot; :
&quot;&gt;=&quot;;
                        String msg = &quot;data[&quot;+i+&quot;] =
&quot;+data[i]+&quot;, was not &quot;+comp+ &quot; than
data[&quot;+(i+1)+&quot;] = &quot;+ data[i+1];
                        assertTrue(msg, bool);
                    }
                }
            }
        }
    </junitCode>
</exercise>
```

END File: /Projects/BlueJ_TA/Code/src/Extension/exercises/SortExercise.xml

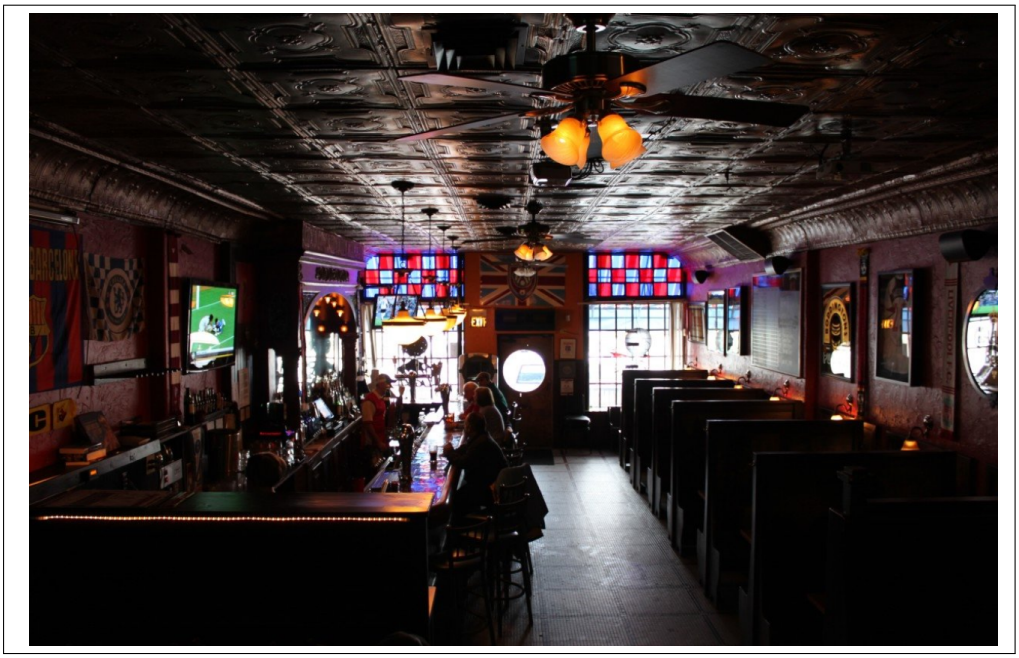
Note: 1 artifacts were ommitted in this abridged version.

Team Building

Summary During the 15 weeks we spent working together, we had the opportunity to meet outside of work and classroom setting for team building events. We took these opportunities to learn more about each other and connect further as a team.

Team Building @ British Bulldog

Description The following artifact consists of a photograph of the restaurant The British Bulldog. The objective of this first team building event was to be able to learn more about each other on a personal level, without discussing the workflow. This event stemmed from a need for the team members to jell together. As a result, team members became more comfortable with working and conversing with each other.

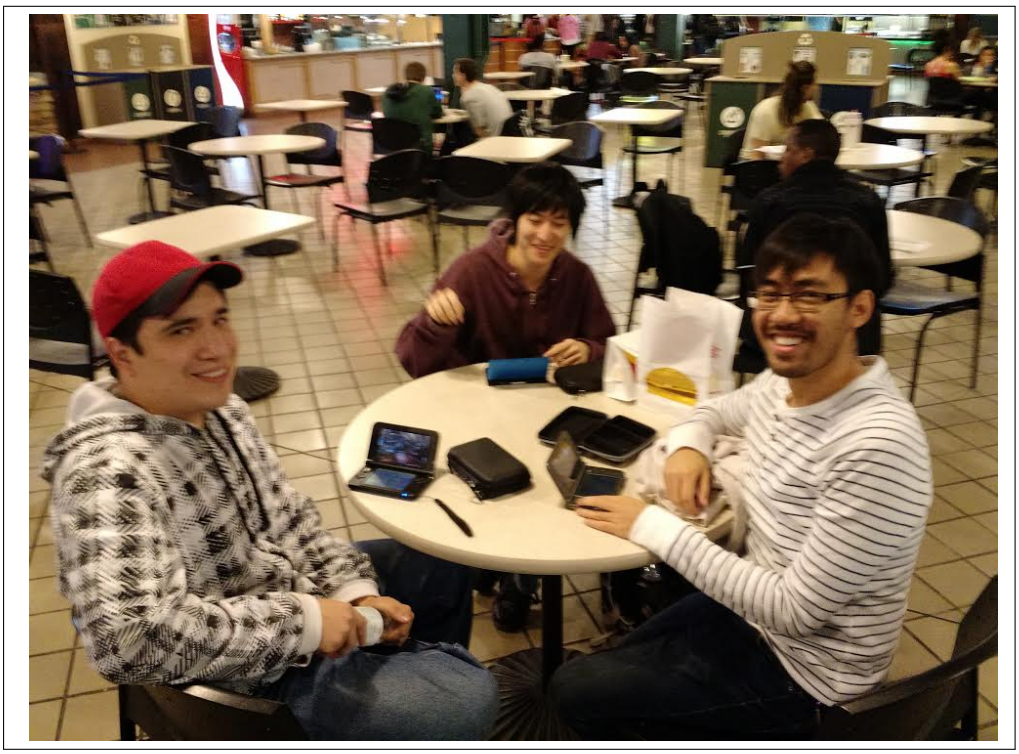


Team Building @ Euclid Hall

Description The following artifact consists of a photograph of us in the restaurant Euclid Hall. The team felt the need to have one last team building event before the final push: working on the final presentation and the team portfolio. Having the whole team come together allowed everyone to relax and enjoy each others company in a non-work setting before crunch time.

Team Building with a Monster Hunter Game Session

Description This artifact is a photograph of the team members playing Monster Hunter together after class. The game lets players work together to hunt large monsters. This video game encourages teamwork and working together, and so this event helped team members jell together, communicate with each other towards a common goal, and participate in activities not involving the main project.

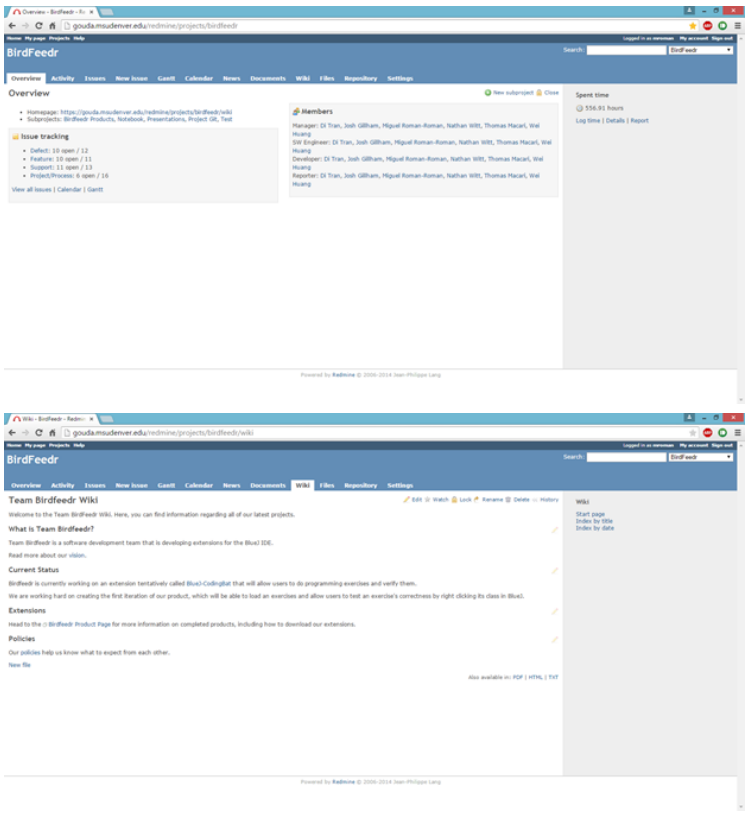


Tools

Summary Our team used a number of tools to facilitate our project. We selected our tools because they were reliable, supported, and we had expertise. There were other tools that we voted down like Git. Although Git is an excellent tool for version control, we voted it down because we did not have expertise and we would have to wait while this tool was set up.

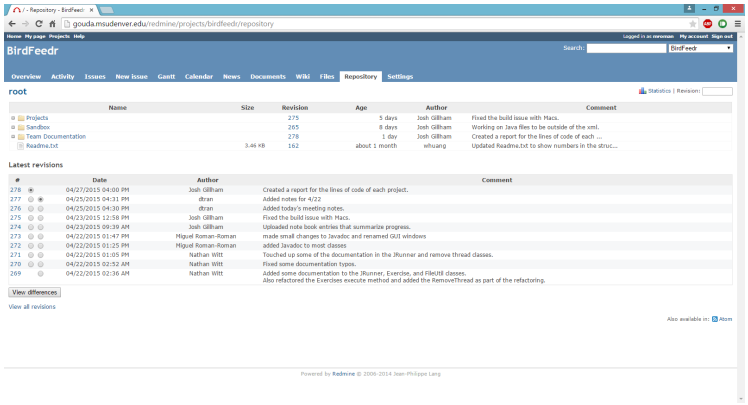
Redmine

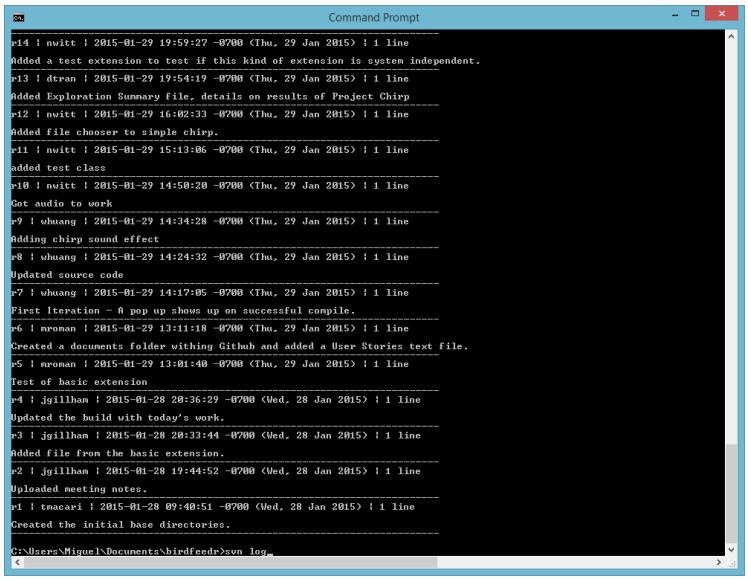
Description Redmine is the web based project management tool used by our team. Team hours, project issues, and wiki page were placed here. This tool gave stakeholders an access point to what we were doing. The following artifact consists of two images of BirdFeedr's Redmine.



Subversion SVN

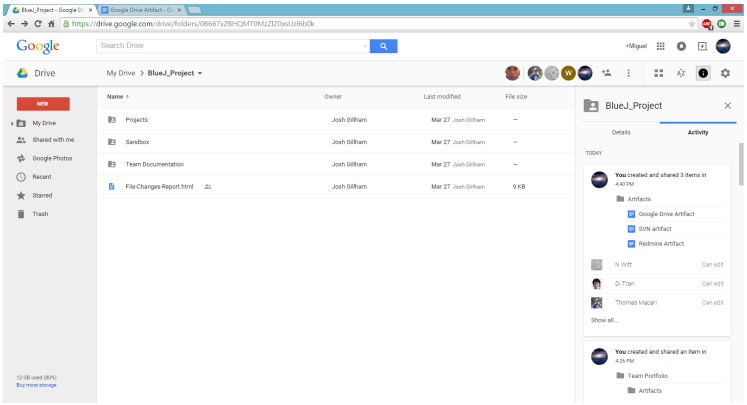
Description Subversion is Team BirdFeedrs version control system, and this repository contains all of our produced work. Since the repository is version controlled, the team was also able to revert back to previous versions of any files if necessary. This tool allowed team members to share the product between team members and stakeholders. The following artifact consists of two images of the SVN Repository.





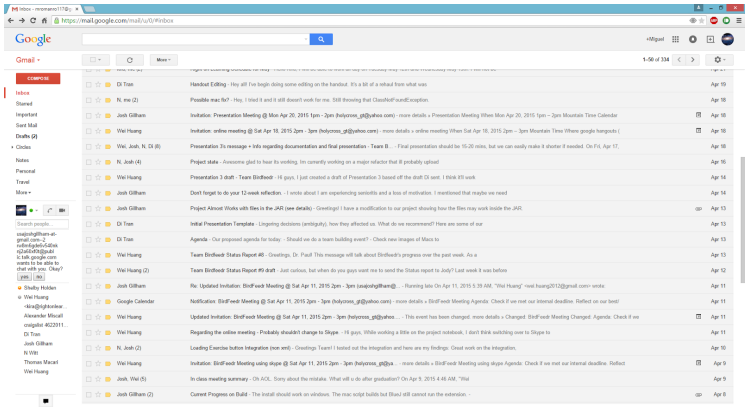
Google Drive

Description Google Drive acted like an additional repository specially used for synchronous editing of documents. This tool helped the team produce necessary documentation while allowing all team members to access and edit the documents simultaneously. For this reason, Google Drive was a very important tool for the team. The following artifact consists of an image of our teams Google Drive folder.



Gmail

Description Gmail is the primary email service used to communicate to other team members off site. Email was used when it would be considered too inconvenient to gather everyone together, and was generally needed to communicate about the project during non-scheduled hours. Since we used several Google tools such as Google Drive and Google Hangouts, using Gmail made it easier to keep everyone together. The following artifact consists of an image of a team members Gmail page.



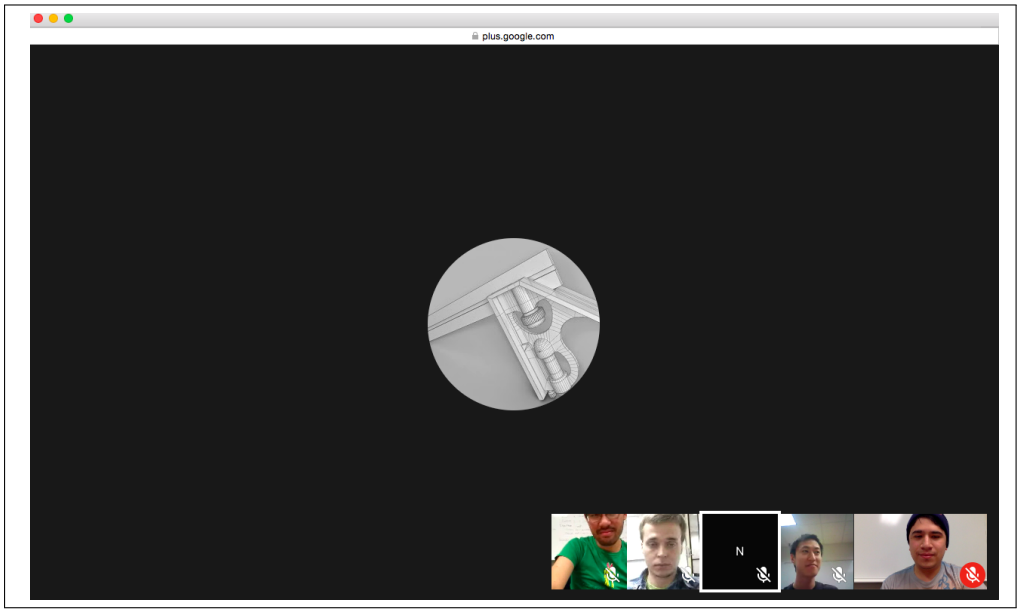
Google Hangouts

Description Google hangouts is an online video chatting tool that allows members with gmail accounts to communicate online.

Some of the features of Google hangouts allowed us to video chat as well as screen share to show each other our ideas and code.

We used Google hangouts to facilitate meetings with each other more often when we weren't able to meet in person on a weekly basis.

The following artifact shows an example of what our hangouts look like.



JavaFX

Description JavaFX is a Java library that helps in the creation and deployment of graphical user interface for a program. BlueJ TA utilizes this library mainly for its own graphical displays. We decided to use this approach because of the simplicity in making and managing the code for JavaFX using available JavaFX tools.

The following artifacts consists some samples of what JavaFX looks like.

BEGIN File:
/Projects/BlueJ_TA/Code/src/Extension/GUI/FXMLExerciseDocumentController.java

---

```
package Extension.GUI;

import java.net.URL;
import java.util.ResourceBundle;
import java.util.ArrayList;
import java.util.List;

import javax.swing.JOptionPane;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.ListView;
import javafx.scene.control.Label;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.stage.Stage;
import javafx.beans.value.ChangeListener;
```

***** Lines 20-90 were ommitted in this abridged version. *****

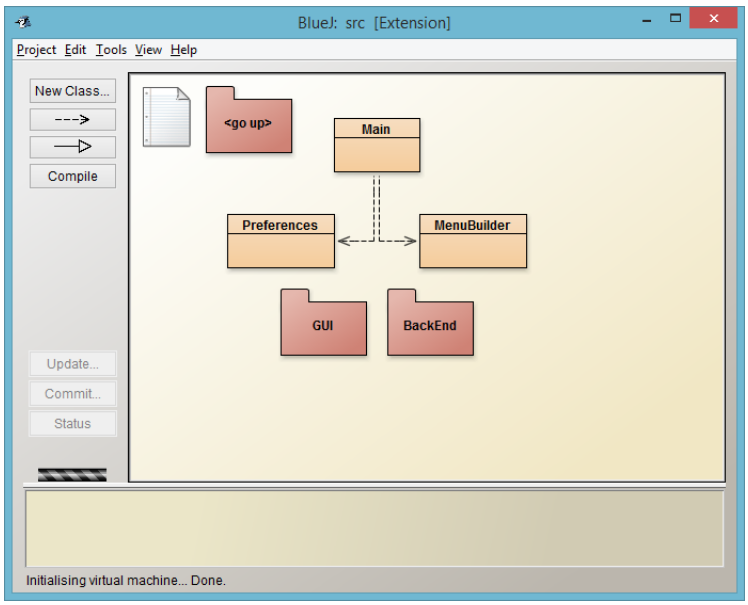
```
        items.setAll(StateManager.getExerciseList());
        list.setItems(items);
        list.getSelectionModel().selectedItemProperty().addListener(
            new ChangeListener<Exercise>()
            {
                public void changed(ObservableValue<? extends Exercise> ov, Exercise old_val,
                                    Exercise new_val)
                {
                    exerciseSelected = new_val;
                    titleLabel.setText(new_val.getTitle());
                    descriptionLabel.setText(new_val.getDescription());
                    String examples = "";
                    List<String> examp = new_val.getExample();
                    for(int x=0; x<examp.size(); x++)
                        examples += "   " + examp.get(x) + "\n";
                    examplesLabel.setText(examples);
                }
            });

    }
}
```

```
END File:
/Projects/BlueJ_TA/Code/src/Extension/GUI/FXMLExerciseDocumentController.java
```

Note: 1 artifacts were ommitted in this abridged version.

BlueJ

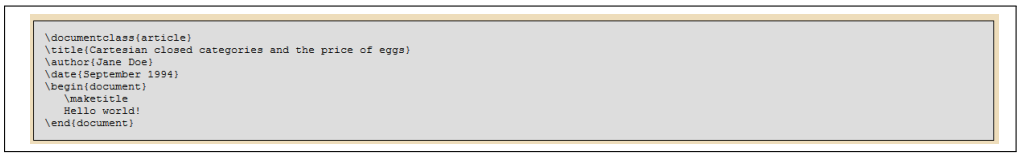
Description The following artifact is a screenshot of the BlueJ integrated development environment (IDE). BlueJ was our teams main development platform and was used early our development to build our extensions. BlueJ is also the platform we developed our extensions for.



Latex

Description Latex is a tool for manufactering documentation. It is widely used for technical or scientific documents. It is like HTML because the author can create content of the document and use formatting codes to control the way the text will be displayed. Thus, it easy to version control. Also, just like HTML it can include other documents like PDFs and JPEGs.

Latex was used in this project to create the project notebook. A screen shot below shows how LaTex code looks.



```
\documentclass{article}
\title{Cartesian closed categories and the price of eggs}
\author{Jane Doe}
\date{September 1994}
\begin{document}
   \maketitle
   Hello world!
\end{document}
```

Final Analysis

In conclusion, we have put in a lot of man hours this semester to produce a product that functions on various platforms. For future teams, we have left the product well documented and in a ready state. During the course of this semester, we have put use the practices of software engineering. Finally, our primary stake holder has give us positive feedback on the various aspects of our work.

We have a product that works on OSX, Windows, and Linux. Our product comes with several sample exercises that demonstrate its features. Plus, the product has the allows users to point the extension to custom exercises.

The project has been left in a ready state and that means that the product's source code has been well documented to enable future teams to understand the code. We have also documented the bugs in our project management tool. External topics have also been covered by our documentation such as retrospective process models, email threads, team building experiences, and so further.

Software engineering practices have been put to use during the course of the semester. For example, we used user stories to focus our team on important features. We also put entrepreneurial practices into use. For example, we created a vision statement to focus our team on a dream for the future.

Our primary stake holder has given us positive feedback on our work. For example, Dr. Paul said that our vision statement was "well crafted." He also liked our team name because he said it was "trademarkable." He also seemed to like our product based on his reaction to the product.

Overall, we feel successful about our work on the project over the course of this semester.

Advice for the Future

Communicate with the professor or stakeholder often. Plan the next step before a group separates. Make sure everyone has something to do while the group is apart. Have a singular vision for the team. Have team building early and regularly. Define the process model early. Exploration projects should be throw away projects. First write down your vision or dream for the final outcome as group. Don't force any team member into one role. In contrast, roles should shift. Let everyone have the opportunity to have an experience in a different role.