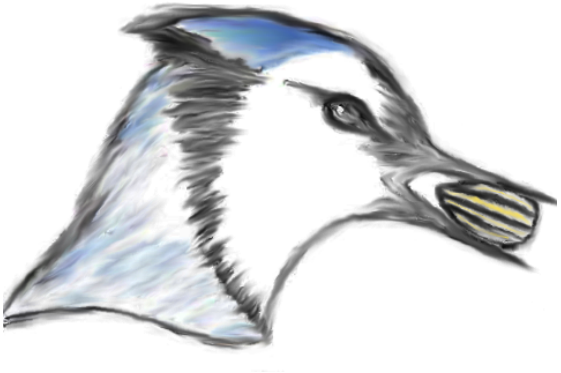


TEAM BIRDFEEDR
Project NoteBook

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.

--
Creative thinking, visual.
Not good memorizing.

Good at math not english.

--
Knows android well.
Notvery diverse in langs

--
Knows quiet a few languages.
Minor in physics.
Training as system eng.
--

Strength: easy to get ahold of.
call of at 4 am.
Professent in java
but not proficient in java.

--
2015-01-21 14:55:43
```

```
2015-01-21 15:26:50
Tran likes video until he heard iOS.

Thomas likes moodle and blueJ addons.
  -Come up with a plan.
  -Fits a certain framework.


Thomas: moodle in PHP.


--
Thomas -- extensions.
       2. extensions.
    bluej or moodle.
Tran - same more bluej
Josh - 1. Bluej 2. Moodle Notifier
Di - 1. Moodle - wants to learn something new and fix moodle.
     2. Bluej
Nathan - 1. BlueJ 2. Moodle Confidence.

Dan -- 1. Moodle 2. Bluej.


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.

Wei: Understands what we mean by the project we want to do.

Wei: Puts his vote in for svn b/c it comes with redmine.

Di: votes the same.

Miguel: Votes the same.

Nate: says its the one he used to.

Josh: Since we know those tools we don't have to spend time learning.

Thomas: has never used redmine.

Wei: YOu just need to knw how to write wiki pages and create issues.

Team names Ideas
Bird feeder
Extendables

Characteristics:
Wei: Proud of it.
Nathan: Something that motivates us.
Di: BirdFeedr

We are choosing Team BirdFeedr w/o the last e because we are making extensions.
We invision our team as feeding the BlueJ community.

Wei: Wants one out of class meeting per week.
Make sure everyone is on the same page.
We can do it more often as needed.
A few key people need to be there.
i.e. only people that are coding the extension.

JOsh: one or more meetings weekly.

```
Di: The weekly the meeting will allow us to touch base.

Wei: class will be mostly teaching.
  Record each meeting so we can keep track.

Wei: we could have a folder in svn for reports corresponding to the project
  notebook.

Thomas: volunteering for managing svn.

  Individuals do not have a notebook but rather a portfolio.

Wei: Update the project notebook after every meeting.

Thomas: The notebook should be in the svn repo.

Josh: There will be a report after each meeting in the notebook.

Wei: Current status -- deciding our extension.
  Everyone should report on that on Monday.

Nathan: Discuss the extension on monday. Clearify.
Miguel Research

FInished svn, metric tracking, team name.
Todo Process model, choosing extension.

Josh: exp for tiem tracking.

Nathan: Suggestion -- Note what you did b4 the day is over.

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.

Dinner Feb 10, 5:30 pm

Wei: Refactoring. Small ext. Easy to maintain.
  Suggestions/analysis and refactoring.
  Support a small subset of refactoring.

Miguel: version control.

Thomas: do a small extension to begin with to give our team a quick
  success. We could do another extension.

Thomas: slightly easier -- ext further languages. Why? Env setup.
  Linux has native support for c++. WIndows does not.
Josh: IDE does not allow opening other files types.
Thomas: The lang tools are out their.

Nate: Visualization. What's on the stack, heap, garbage collection.
      Visualize multithreading deadlocks.
      IT would help people in BlueJ learn.

Di: Simulation also. Harder, but, more helpful.

Josh: Clojure IDE got shutdown.
   Refactoring was feasible.
   Like gen automation testing.
   Like simulation.

Everyone likes the idea of starting with an easy project.

Thomas: clean room dev. Find a bug start all over.

  COT -- specialized XML made by the military.
2015-01-26 15:01:25

2015-01-26 15:19:03

```
Thomas: Two parrallel teams.

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


BEGIN File: /Team Documentation/Meeting Notes/note20150128-Team-Meeting.txt
```

```
2015-01-28 14:09:40

Wei: wants waterfall b/c of the documents

SVN REPO

  Chirp -- project chirp source code.
  Github -- github extension.
  Documentation -- Notes.
    Meeting Notes/ - duh

 Sandbox/ -- individual information, source code.

--

Task: maybe get admin access.

Di & Wie: wiki is about the introducing someone two the project.

Wei: start working an assess time on project.
  Use to estimate.

Wei: next week earliest we should finish Monday.
  Next Next Wednesday.\
```

```
Di: Construct policy document.

Josh: Expected results for sat.
Wei:

Jody: take smaller steps.
Agile wants to get things running right away.

Water fall -> gather requirements, design, code, maintain.

Iteration should be one week.

2015-01-28 15:00:49
```

```
END File: /Team Documentation/Meeting Notes/note20150128-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150131-Team-Meeting.txt
```

```
2015-01-31 14:07:06

Chirp: met and implemented the example extension.
  Created another extension to pull up JOption Pane.
  Ran into a small problem with windows.

Github: Release planning and iteration planning.

---
Wei: go over what went well and what did not work well.


  Peer programming worked well b/c they could get information
    accrossed quickly.

Di: Liked how we were iterating.
   Felt like we were making progress until they hit user directories.
  More design. How to design the extension in general.
  Spend more time on design.

Nathan: Liked the progress. Jumping into it.
  Did not like the slopping code.
  Spend more time cleaning the code up and organizating it.


Josh: where did you make the most progress?

Di: outside of class. Met the day after.


Josh: I liked learning about XP.
      I liked learning about Refactoring, JavaGit, JGit.

Miguel: Could have spent more time on the project.

Wei: More time outside of class.
```

Thomas: We spent time in class working on the extension.

Josh: anything else.

Wei: I don't think there is anything else.

Questions:

How is the process model of each subgroup shown in the notebook?

Josh: We don't want to wait till the last minute to work on the notebook.

Wei: Keeps a personal notebook and records decisions.
Push that into our documents folder like a project journal.
Subgroups keep their own Journal.

Josh: Di you want to add to that?

Di: How are we going to document this?
If we are can ship jar files then add the thinking process for each.
It would be better to work on project notebook.

Josh: Miguel anything?

Miguel: Nope.

Nathan: We are not following a process model.
Getting to know the process models.
Process model is not concrete.

Wei: Closest process model is closet to Agile.

Nathan: Purpose of process model is to complete a project.
If we spend more time on process model than on the actual work
then it is like a leach.

Wei: Choose the process model that works for us.
It helps us when we don't know what to do.

We start working and choose the process model that fits us.
We are trying to be philosopher.

Josh: I think this has

Josh: Let the project Chirp be success and Git be a failure.

Di: Project Git got things accomplished with the user stories.

Wei: Prj Git wished they got their hands dirty.
Now one group has experience Project Management and other group has
experience in getting dirty.
Now one group can rely on the other.

Di: Wants to ship the extension to everyone in class.
Get practice on learning how to ship.

Asssume we will do a presentation on Monday.

Nathan: Good controlled environment.

Wei: likes Di's idea b/c we get experience w/ maintainence.

Thomas: It touches on the practice of the deployment phase.

Nathan: its like a focus group.

Miguel: Are we suggesting presenting the Chirp extension?

Wei: Yes, but also deployment.

Thomas: Idea: Since prj Git is not done and we had a different style of iterations. Prj Git continues its iteration.
Both subgroups are working a making progress forward.
It allows us continue with our iteration style.

Wei & Di & Nathan & Miguel: Agree.

Nathan: Another week of expereience and we can do risk analysis later.

Josh: We formed tasks from the release planning draft.
We need 10 hours together.

Di: Primary problem getting time together.

Wei: Divide into peer groups.

Josh: What's peer groups? is it like pairs?

Wei: Kind of.

Wei: AFter project Chirp finishes distribution to address the time issues.

Josh: Would we continue or help project Chirp?

Josh: To rest of the group: can we find time to meet?

Thomas: Can't do Sunday b/c of super bowl.

Wei: Offers his help. Free tomorrow.

Josh: Miguel can we match schedules?

Miguel: Saturday today and Tuesday.

Josh: Can we do this afternoon and Tuesdays afternoon.

Miguel: That works.

Josh: Ill send you guys a calendar invite.

Wei: Is willing to help.

Wei: Will deal with practicing, maintaining. and presenting in front of the class.

Thomas: One suggestion for Chirp: send an email to Jody and the class to take time for a presentation.

Wei & Di: Agree.

Wei: That would give them time to install BLuej

Nathan: We have 3 months left.

Meeting adjourned.
2015-01-31 14:58:21

END File: /Team Documentation/Meeting Notes/note20150131-Team-Meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150202-Team-Meeting.txt

---

2015-02-02 14:29:59

Good idea to split into groups.

Di: Keep track of our accomplishments to present on Monday.

Wei: over 5 mins is too long.

Di: Ask Jody about putty BlueJ on the Wikipedia.

Wei: Appoint a vice-speaker to jump in to support th speaker.

Nate: Make one of presentations based on two of subprojects.

Project chirp can continue on more features.

Thomas: Each presentation needs to have tips and tricks.

Josh: Project Git needs to establish a common Metaphor.
2015-02-02 14:47:33

END File: /Team Documentation/Meeting Notes/note20150202-Team-Meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150204-Team-meeting.txt

2015-02-04 14:19:46

Status Reports:

Di cant make it Saturday

Josh" We can have two demos and use the 2nd to catch Di up and get things done.

Wei: Set presetnation date.

Josh: We need topic.

Thomas: Live demo.

Miguel: Could waste a lot of time.

Wei: have one machine to do the demo.

Di: Does everyone agree?

Wei: Having a demo at the same time as the demo means distraction.

Thomas: Discuss key points of why this was useful and how it applies
to management.

Josh: idea an ext that helps visualize teams working together.

Decision: Do Sat at 2PM and update me with a meeting summary (Di).

Josh: I will start the thread with the food decision.

Thomas: Ackomplishments, Current, Future plans.

Josh: How do we provide the other group with value?

Wei: Tell them what we are working on?
Summarize decisions, what we did, how the product.

Nate: Will it really be that interesting to them?

Wei: More like an overview..

Thomas: Discuss what our big project will be.

2015-02-04 14:35:55

2015-02-04 15:43:56

Objectives:
Think about presentations next week.

Large product

Wei: we could modify the product but there is a huge risk.
We make a regular extension.

```
  We could

Josh: Our extension has its own editor.

Nate: We would have to modify the editor and BLueJ does not allow.

Wei: An option which opens a new editor showing the before and after.
\
Josh: We could use their source code and include that as our editor.

Nate: Does know if we can modify the events.

Wei: idea: api browser option.

Di: that will value.

Wei: Do we want to implement small extensions with seperate teams.

Thomas: continue Project Git during this weekend.

Goals:
  Decide the next step or project.

Thought keep teams divided to accomplish more.
  Decide presentation.

Miguel: wants to meet at 3 pm Thursday on google hangouts.
2015-02-04 15:54:34
```

```
END File: /Team Documentation/Meeting Notes/note20150204-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150207-Team-meeting.txt
```

```
2015-02-07 14:06:05

Wei: Idea: block code generator.

Josh: API offline documentation.

Wei: A visualization of Java source.

Josh: Helps people learn.
 Use abstract syntax tree.

Thomas: The current ideas don't sound bad.
  Idea: find bugs.
  Implement a find a bugs extension

Wei: we might have to maintain the project even after.

Nate: the virtual display requeries deep research.

Miguel: Idea: Intellisense.
```

Wei: Problem: modify BlueJ source.

Josh: BlueCLJ. IDE supports clojure.

Wei: Do risk analysis on each of the ideas?

Thomas: Jody likes tangible proof of work done.

Josh: Project Chirp and Project Git give status report together.

Josh: Get consensus on our Research presentations and present next week.

Nathan: Set the date and topic.

Josh: Presentations about how to pair people people together and assign roles.

Wei: So that is team building.

Nate: Thomas said pick a topic and go with it.

Josh: Project Git has accomplished this week's goals and surpassed them.
We implemented the Git Commit and Init features.

Miguel implemented the clone and push features.

Wei: Project Chirp has been researching various topics.
Project Chirp will gather information and get and idea for the presentation.

Josh: Project Git will try to polish

Wei: Don't polish Project Git and just document everything.

MigueL: project Git is in the way of the bigger project.

Wei: The team meeting will be on Tuesday and when we get the big group together.

Josh: I don't want to talk about work in our team building meating.

Wei: I agree.

Decisions:

Have a idea list by monday. Wednesday finalize.
Project Git wraps up.
Presentation researching.
Next presentation.
Schedule reminder on Monday.
--

Josh: Some people work better solo.

Nate: Pair programming increases communication.

Wei: 2 conditions: things get done and people know what you are doing.

```
   Document decisions made alone.
   So people understand why.

Project Git will try to remove debugging code.
 and produce a status report for monday.
Josh will produce a team wide status report.
2015-02-07 15:26:57
```

```
END File: /Team Documentation/Meeting Notes/note20150207-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150209-Team-Meeting.txt
```

```
2015-02-09 15:03:46


Thomas: likes a sweet of extensions.
   We can pick several extensions.
   Everybody gets to play everyrole.

Agenda: talk about the final project ideas
  talk about the presentations.


Thomas: We gave a presentation last semester about the
  principles of waterfall and then sprinkled clues about how we used.

  That is, here is what it is general and here is how we applied.

Di: Generalizing makes ideas useful for others.

Thomas: the process is to pick a project and try and make it useful to others.

Wei: In regards, to JP Git project suggestion
  is everyone in favor of a suite of extensions.


Thomas: want two more extensions to call it a suite.

Di: Ext in the suite don't have to be related.

Wei: We can do the API downloader idea.

Thomas: an offline API browser.

Nate: Fine w/ suite.

Josh: the pro of a suite is less pressure on the idea choosing.

Nate: con we never make anything useful.

Thomas: Reflecting on the time line to Git project.


Wei: If we do the suite then the Git Project should be documented regardless.
```

Would like everyone working on the presentation at once.

Thomas: RPI it.
Have team members do a skit.

Nate: RPI would make more content.

Thomas: Goal -- value for us and audience, but, I need to enjoy it.

Wei: have a guy in the audience

Thomas: show pair programming vs chief programming.

Josh: we need to top down design tasks.

Nate: We need rehersals.

Di: people need to read lines.

Thomas: the people reading the lines off the slides could be scripted.
Some people

Di: We need a structure planned out.

Thomas: Some people could be on improv.

Di: we need to be in the same place for a rehersal.

Thomas: we could let JP know we need to leave the room for rehersals.

We know class periods will have everyone here.

We decide to use imprav on what we learned.

We decided not to go to Comedy Works.

We decided on the British Bulldog.

Wei: If we did not do suites then we would have to put project Git on back burner.

Wei: If we did suites then would could keep working on Project Git.

Thomas: Creating an installer in net beans is easy for net beans.

Wei: A con would be taking time to make the installer on suite.

Nate: pro for no suite. Focused making a polished project.

Thomas: con for suite. Scope. Might choose something too big.

Thomas: Either way, tomorrow is a prep for wednesday
Wednesday and scope the idea and do research.
Get an idea for what would be involved.

Miguel: would we still be in sep rpojects.

```
Thomas: multiple lines of of production.

Josh: there could be a task list.

  One person needs to make sure that there is enough HR staffing for
    that project.

Thomas: Let there be one founder that makes sure that project
   moves forward.

Nathan: Butting heads.
  For example, if two projects need hours and there is not enough hours
   to go around.

Josh: We could choose project proirities to begin with.

Wei: set a cap on the projects that can be worked on.

Thomas: this may not be a problem if they are staggered.

Wei: The suite may require more documentation which could be con.
   It could provide more ways to track our project..

Wei: Keep work out of the team building.

Josh: likes suite + more creativity, I can keep working on suite.

Thomas: Everybody gets what we want.

Wei: Likes the installer.

Nate:
  Suite rigidity can be a curse to projects.

Miguel: Votes suite for project Git.

Thomas: We should research intentions.
  Give a rough estimate on what it would take.
  Think about what we need for presentations.

2015-02-09 15:54:59
```

```
END File: /Team Documentation/Meeting Notes/note20150209-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150211-Team-meeting.txt
```

```
2015-02-11 14:04:19

Wei: Pair programming is the topic of presentation.

Wei: show a bad programming approach and show pair programming as the solution.

Nate: Focus on pair programming and show how not to do it.
```

Wei: research what makes pair programming effective.

Nate: sometimes its faster just to program it solo.
Long term, it could be benefitcial.

Wei: Everybody participates.

Nate: Two people present, Two people do the bad, and two people do the good.

Wei: the first person goes last to bring familarity.

Wei: show the two good helping out the two bad.

Di: some people could be doing research. Others could be presenting.

Wei: have project git working and add one small feature to project chirp.

Wei: Do a Saturday group meeting.

Next week we will be rehearsing.

Nate: its on the 25th

Di: yes.

Wei: Monday half present research, other half rehearse it.
and fix it if we don't like it.

Thomas: we need to define a cut off point for project Git.

Josh: Instead of cutting Git off, we could just plan one iteration at a time.

Wei: We could be the QA team.

Nate: make project Git worth while.

Josh: We would need to write user stories and priority.
and pick the highest priority first.

Josh: we could create QA tasks and send them over.

DI: we could check on things using our checklist.

Josh: could you document the checklist.

Miguel: the next prj should be decided by everyone.

Wei: We should get more input from our side.

2015-02-11 14:32:22

2015-02-11 15:23:39

Wei: What is our market?

Nate: Go talk to CS1 students about what they want.

Wei: ask JP what he wants.

Wei: mission statement: help teachers.

Di: make BlueJ easy to use and make Java easy to use.

Range user just picked up BLueJ 1 to 2 years.

Josh: should we create an umbrella for new project or existing projects.

Di: swift has a playground feature.

Josh: We could start next week on a mission statement.

Mission Statement: Encourage students and teachers to use BlueJ.

Wei: Goal 1: talk to teachers and students to see what they need.

Di: Jody wants a vision statement.
We need to figure it out.
What will we development.

Help make Java easier to use by enhancing BlueJ.
We would have to cease dev.

Make java easier to teach and use using BlueJ.
When someone is in CS1 one, they don't lern version control.

Thomas: expand mission statement: Provide tools to help Java.

Miguel: What would be the first thing.

Wei: ask teachers what they want.

BlueJ has intellisense.

2015-02-11 15:57:31

END File: /Team Documentation/Meeting Notes/note20150211-Team-meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150214-Team-Meeting.txt

---

2015-02-14 14:02:57
Product vision on the agenda

General consensus to shelf project Git

Wei: project git has lingered on unintentially.

Miguel: shouldn't we come up with a vision statement and see if Git
fits under that.

Wei: We need to work on our presentation.

Nate: presentation and vision research could go together.

Josh: we should be prioritzing by grade value.

Wei: so the higher priority should be on vision.

  Monday we could spend half on the vision
  and the other half should be on the presentation.

Di: so we should spend the rest of the time on the meeting on the vision.
  lets move towards the vision.

Wei: Extensions that faciliate learning for students and instructors.
  that covers almost anything we could do on BlueJ.
 Alternative: ext that specifically facilitate students and instructors.
  pro -- we can ask students here and teachers.
    so more contact with the client.

Miguel: likes the 1st one b/c it gives us more freedom.

Wei: A broad statement could come back to bite us b/c
  we don't have a sense of purpose..

Miguel: if we get too narrow.

Nate: groups last year had trouble with taking control of their own projects.

SOURCE:
http://www.changefactory.com.au/our-thinking/articles/the-components-of-a-good-vision-statement/

Nate: we have our own vision statements. We just need to align them.

Di: Do we all want to support the vision statement of helping students learn.

Wei: I am.

Nate: I think so. Teaching something you don't know and making something
  you know easier.

Di: BlueJ is meant as a learning experience first and a coding experience 2nd.

Nate: Learning approaches should be like training wheels that people stop using.
Wei: We could make a something like a paper clip from Word.

Di: one of the major obstacles is the fear that I am jumping into something
  I don't know.
  Could we make an ext that overcomes that.

Wei: The idea of coming up with a presentation similar to
https://scratch.mit.edu/#
  Similar to that but different.

Josh: lets create a document to help us when we get confused.
  One of the sections would have a vision statement.

Wei: would we include decisions in that.

Nate: maybe we failed at the coming together.

Di: a vision should give people a picture.
  Present tense.
  Emotion.
  Picture.
  Expression.
  Do not confuse with business goals.

Josh: Students who love java haha.

Wei: Simple. Evokes emotion. Summarizes with powerful phrase.
   We make extensions that do things that are awesome.

Di: Trying is putting us in the right mindset.

Nate: We want to make a product that when used by a CS student will enhance
  their understanding.

Wei: We make extensions that make learning java easy.

Wei: I dream that our ext will make learning java easy.

Di: Lets stay away from using ext.

Wei: produce software to make learning Java easier.

Miguel: Elicit the teaching of java in CS students.
 elicit = draw out.

Di: I will describe a pic in my head.
  CS students approach CS students with an open mind and find it fun and
rewarding.
 Change the image from something imposible to fun and interactive.

Josh: introduce more creativity.
*Nate:

Nate: programming college is very rule based.
  I had fun being explorative and creative.

Di: My drive was to make something and build.
 The emotion was to build stuff.
  but the rules get in the way.

Nate: creative programmers are highly successful.

Josh: We would like to inspire creativity in CS students.

Wei: Inspire CS students to become better programmer.

Di: We strive to remove to remove barriers of computer science to make it
  a rewarding experience.

2015-02-14 15:03:15
2015-02-14 15:07:54

The vision gives us a broad scope and direction.
  It keeps us from getting confused.

The project or business goals help accomplish that vision.

Di: Leave everything we have worked on in a ready state or continue working?

Conitnue working.

Meeting break.
2015-02-14 15:09:28
2015-02-14 15:12:26

Wei: the last 10 minutes we should prepare the atatus update.

Wei: we need proper justification for vision statement.

Josh: each barrier b/c a project.

Di: i.e. Java syntax, compiler errors, bugs that show up.

Nate: unnatural incorporations of the java syntax.

Di: Differences b/t the way holiwood portrays CS and reality.

Josh: students can't easily make connections b/t statements and behavior.

WeI: remove the end: rewarding*

Di: wants to keep rewarding in.

Miguel: implies CS is not rewarding.

Nate: we know a secret about CS.

Josh: We strive to encourage education by give students over their learning experience.

Josh: We seek to make programming a fun and interactive learning experience.

Wei: Project idea: jargon library explains typical constructs i.e. int i = 0;

Miguel: Simplify the teaching of computer science to students.

Nate: Faciliate learning why? That is where the emotion.

Josh: the current state is that most people drop CS classes.

Josh: help people reach their dreams of computer science creativity
  by removing the barriers that make learning programming difficult.

Wei: to help people reach their dreams of computer science creativity by making the barriers of programming less difficult.

Wei: This is what we want to do and this how we will do it.
  What we want and how we want.

Di: Lowers the barriers => we will make you easier.

Miguel: You could also think of it as we lower it all the way.

Nate: One makes the programmer stronger.

Di: We do have the power to make it easier to learn.

Nate: if we want them to learn then they will eventually need to tackle that
   those goals.

Wei: 2nd a little. We remove thE barriers and not their barriers.
  make a reference to them.

Josh: what are the state of the barriers to programming.

Nate: we help them navigate the barriers to programming.

Di: the tools help them overcome permentantly.

Wei: I pick miguel's.

Di: Should we be allowed to change this later?
  If no then put weight on this being the right vision statement.

Di: Lets get everyone's opinions.

Josh: I think its down to the precision we need.

Miguel: It encompasses everything.

Nate: Its good. We never not want to make a tool

Di: Based on the above reasons, I think its good.

Vision Statement:

To help people reach their computer science dreams by providing tools that help
overcome barriers of programming.

2015-02-14 16:02:39
2015-02-14 16:02:55

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

Status update:

Nate: if we had a policy doc then people prefix a tangent with "this is a
tanget".

Di: a tanget should be done.

```
Status update:

Created our vision statement.
   Our mission statement is ...

Brought unity back to our team.

Talked about issues dividing us.

   philisophy -- con of splitting up is the difficulty rejoining.

New policy for tangents.

Closing out project Git.
  Nate could explain why.
  Nate -- git was not a bad project.
    trying to run too fast.
    the project was making the decisions.

New policy about final decisions.
    go into a final decision with consensus has to unanimous.
    to break a decision we need unanimous decisions.

Create a document to help us when we feel confused
  that includes the vision statement.
--

Update Thomas.

Wei: make decision section in the document (vision document).

Next class first half on presentations and 2nd half on project topics.
  2nd half after the break.

2015-02-14 16:29:45
```

```
END File: /Team Documentation/Meeting Notes/note20150214-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150216-Team-meeting.txt
```

```
2015-02-16 15:13:54

Presentation:

Two narrators.

Two role players with the bad example
Two role players with the good example.

Thomas: CS larp live action role play.

Miguel: will have slides.
```

Wei: slide with peer review.

Nate: narrator on slides.

Thomas: slides will be the work of narrator.

Wei and Di will be bad larpers.

Nate and Josh are good larp.

Thomas: have one narrators.

Josh: we need points.

Thomas: each group has a set of points.
  How can situation be fixed.

  or What is good.

Di: Driver-tweeter.

We have to be prepared for no questions.

Thomas: You guys come up with ideas about bad programming.

Nate: the presentation needs a heart.
 a core.

Wei: Should presentation fit inside our vision as well?

Next mondays expecation.
  Get the main points read for next Wednesday..

Nate: concept of reuniting.

Di: bring things together.

Project topic.

Di: Think about our product.
  Lets pull out past ideas and see if they fit.

Thomas: Will read the ideas.

  Other code. nay
  Refactor. nay
  Virtual under the hood display: yah.
  Updated UI: nay b/c of class card extension.
  Block generator: nay b/c it could generate dependency.
  JUnit starter: nay too advanced.
  JUnit statistics: nay too advanced.
  Open GL integration: nay Green foot support.
  BlueJ tutor: yah reads the code to the user.
  Android cell phone exporter: nay not education.
  Change logs: nay not a learning tool.

```
JP: JBoxes. animates java execution.
  BlueJ tutor -- pick the things that our problematic.
    point out if variables are instance and class.
    Scope and domain would be difficult.
    UI addressing that aspect.

2015-02-16 15:53:17
```

```
END File: /Team Documentation/Meeting Notes/note20150216-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150218-Team-meeting.txt
```

```
2015-02-18 14:02:42


Wei: will not be Sat.


Nate: Pair-programming novice-expert the expert does all the work
  and does not contributing to the code.

Presentation is Wednesday of next week.

Josh: risk: giving them what they don't need.


Nate: differences b/t code review and pair programming.


Wei: Shows here is how u do it and here is how you don't do it.

Josh: the title could be the do's and don'ts of pair programming.

Di: Split the presentation into blocks of good and bad.
  Here is the example
  What was wrong.
  Here is the example.
  What was good.


Di: want to include novice-expert.

Wei: Show the novice-expert as a pitfall.
  The novice is not contributing anything at all.

Di:
  intro topic
  demo
  lesson
    ask audience
    explain what was good or bad.
```

Di: expert-expert is another scenario.

Wei: con -- one complete takes over.
don't think outside the box.

Show good fib and bad fib.

Di: novice-novice can be the last block.

if they both are too poor then the COACH has to step in.

Learn together. Confidence booster.

Nate: How can novice-novice go right?

Wei: they can confirm right answers.
they might not be able to figure out hard problems.

Give them something easy to do.

Unimportant.

There must be a person to answer questions.
pro -- gets them into the project but does not compromise the project.

Josh: they could distracted i.e. watch videos.

Di: How should we do conclusion?

Wei: reitterate main benefits of pair programming. Key points from each pair type.

Di: encourage the audience to try it out.

Wei: all questions reserved for the end.

Nate: Keep the title of dos and don'ts

Josh: pros and cons.

Di: its about showing benefits and disadvantages.

Wei: keeep that the vision.

taking a break.
2015-02-18 14:59:10

2015-02-18 15:07:53

Wei: give hand outs at the end.

Josh: artifacts?

Di: handouts, props, table, two chairs, one computer,

```
Josh:

Nate: the code would be a distraction.

Wei: use the table center class room.

Di: novice-novice no one knows what they are doing?

Nate: tangent: found benefit more comprehendable.

Wei: novice-novice they know how to program but no one is good at it.

Di: narrators have the most script
   people in the demo have more generic scripts.
  improvised.

Rehearsing

Nate: bad group should always go first.
  so that the audience can learn that way

Di: make that block-consistent.

Di: each partner should be actively communicating with the other.
  bouncing ideas off each other and learning.


Note: Di and Wei actively work though the code with each other.

Wei and DI: finish the presentation by Sat.
   Write out what we need to say.

Di: I can be prower point own.
 we need the script.
 We need a presentation.

  Narrators should be in charge of their own.

Di: I will finish the structure by saturday..

  By monday the opowerpoint should be conven
2015-02-18 15:54:33

2015-02-18 16:05:58

2015-02-18 16:10:24
```

```
END File: /Team Documentation/Meeting Notes/note20150218-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150223-Team-meeting.txt
```

```
2015-02-23 14:05:49
```

Todo:
ANimation of slides
QA of artifacts i.e. artifacts.

Agenda:
-Presentation
-Projects.

Props:
-Good pair uses Nate's PC.
-Handicap desk.

-Good pair left. or house right.

Projects:
Open GL is too general: yah pending futher investigation.
Tutor: nay b/c of scope pending Nate's plan B.
Mobile: nay does not overcome barrier's to programming.
Git: nay
Explore logs: nay b/c we not helping people reach their dreams.
-Does not help people overcome barriers to programming.
Method Executor: nay b/c its not inspiring and its more like a test tool.
Big O analyzer: nay
Multi-thread: yah visualizing threads would help people with that barrier.
Java interpretter for BlueJ: yah.
Animation clip that provides support: yah.
pending further investigation.

Nate's Plan to reframe the vision statement and work on Git and make a really nice extension.

Accomplishment: Filtered out projects.

Plan:
Tuesday do a little research on each topic.
Wednesday choose a project.

Individual:
Assignments everyone comes up with a informat 150 word report on their favorite topic.
2015-02-23 14:43:27

2015-02-23 14:55:40

On Wednesday, talk about presentation results, talk about new presentation, and decide project.

Presentation time

Miguel: time presentation.

Nate: might take shorter.

2015-02-23 15:01:36

END File: /Team Documentation/Meeting Notes/note20150223-Team-meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150225-Team-meeting.txt

---

2015-02-25 15:22:05

Di and Nate have been pulled out of project Birdfeedr.

They need to determine the feasibility of using an antanna
  to ping a cheap.

Thomas: don't bother them, but, keeping them in the loop.

Josh: Continue to send meeting summaries.

Thomas: Send weekly summaries to JP and them.

Thomas: when do you go on the clock for that.
  So you are not going off the clock now.
  Is there anything you would like to present.

Josh: On the other hand, you guys could participate in the meeting.

Di: Offer informal reports on project ideas.

Nate: be sure to transfer artifacts.

Di: Lets talk about ideas.
  Some ideas talk need access to the editor.
  Clippy requires the editor.
  Big risky. Out of scope.
  Likes interpreter.

Thomas: Build hooks b/t IDE BlueJ and Java REPL

Josh: use an interpreter.

Di: likes Thread visualization.
  Not sure if he likes OpenGL.

Thomas: JP did not like OpenGL.

Di: Likes picking up Git, but, we would have to invoke final decision policy.

Wei: really likes the interpretter.
  and thread visualization.

Nate: Those two ideas. Interpretter.

Miguel: Can we use the technology?

Nate: need to make GUI.

2015-02-25 15:34:22

Nate: what if it is bad choice?

Thomas: then the feasibility report fails. We step back and rechoice.
  How do manage and minimize these choices?

Nate: Interpretter would be easy and risk would be user interface.
    Thread visualizer needs GUI access to runtime, object, locks.
    Need access to the runtime machine.

Thomas: Java has hooks into event watchers.

Nate: If we did go back to the git then people have experience.
  Add all features of Git Pull, commit.
  Consider available time minus presentation time.
     minus people's weekly time.

Thomas: Visualizer. likes idea of heap and stack.
  Differentiate b/t stack, heap visualizer and thread visualizer.

Wei: What if we have interpretter extension that can visualize threads?

Thomas: We got this done quicker then lets add to that.
  Seems like a negative.
  Lets add to that.

Wei: each one could be an iteration.

Thomas: that comes out of our analysis.
  Why don't we say this?
  Add to the feasibility an analysis of adding a thread visualizer.
  Develope reports on all three.

Josh: Each person takes a feasbility.

JP: make sure you get a validity check with me.
   I can be talked into things

Thomas: Send ideas JP way then feasibility.

Ideas:
  Interpretter.
  Thread Visualizer.
  Stack visualizer.
  Offline Java docs.

JP: Sounds okay.
  Nothing exciting.
  Propose it in a way that contextualize.
  How is this helping the average programmer.
  Visualizing threads is not bad.
  What do early students struggle with.

```
  They don't know how to do design.
  They don't know what quality code.
  THey have trouble debugging.
  Tools identifies nouns, verbs, attributes.


Thomas: Send small questionaire to see what they struggle.

Thomas: making a questionaire.

Miguel, Wei: talking to a variety of people to find out why programming
  is difficult.
2015-02-25 16:03:59
```

```
END File: /Team Documentation/Meeting Notes/note20150225-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150301-Team-meeting.txt
```

```
2015-03-01 16:06:17


JP wanted something to facilitate pair programming.


Wei survey results:

Sample Size: 11

90% said syntax.

What they did.

Evaluated. Integrated psysiology major.
  Hardest part syntax and functions.
  i.e. what was static.

Trend talking to someone else about programming helped.

Talking to a tutor.

Object-oriented to functional programming issues.

--
Used different questions for professors.

What makes it easier for your students to learn?

JP said plug and play like green foot and scratch.
  i.e. program the cars and see them move around.

Shultz -- role playing and acting out.
  Have a network program that has students work on the same code
  Over the monitor split screen.
```

```
  That aligns w/ JP idea.

JP Not building good enough coding habits in CS2.

  Facilitate better programming practice.

Basides syntax another common issue is data structure
  visualization and algorithms.

What helped is talking to someone about?

--

Miguel: Idea: make an ext that makes it easier for them to understand compiler
errors.
  Explain where it would occur and what happened.

--
Wei: we could add that to a larger ext called the facilitator ext which
  does everything we want it to do.

---
Thomas: Up for coding anything. Project.

Wei: Lets make one big extension that has everything.
  We have a main ext and we build classes into that ext.
  We can work in parrelel.


Wei: should have another survey?


Josh: are we confident on the decision?

Wei: We have surveys and online backup.

    We should grow a project.
    Start with something super small.
    This is a cool idea lets build on that.

Josh: So we build an iteration and get feedback.

Wei: We build a demo
   and from there keep adding.

Wei: There is two ways to go about this
   One big extension that would help people.
   Or we make a list and show it Jody.

Josh: one week project.


Wei: THe next iteration is this.


Team break:
```

```
Josh: I would be cool w/ a project idea.

Everyone: Lets compile a list and send it to Jody.

Ideas:
 Thread Visualization.
 Stack/heap visualization
 API Doc.
 Explain errors.
 Split screen network program.
 Pair programming facilitator
 Pair programming timer.
 Ask teacher.

Students love the idea of something that helps them find what they.

 Thread and heap is good the students want to know what they are doing.
   API Doc.
   Explain compile errors.

 They liked typing in english and having the program split something back out.
 They liked the error.

Miguel: SInce JP is manager then he must approve idea.

Wei: We got through the students then JP and waste 2.
     We go through the JP then students and they don't like it and we waste 2
days.

Wei: Go with Miguel. B/c impatience. Too much thinking. Not enough doing. Tired
of talking about it.
     If we talk to the students and they shut it down then it would be weird.
     If the students back something up that is we have evidence then there is
no way he could shut it down
        unless the idea is implemented.

Thomas: frustrated b/c we are spinning

Talked about the results from the students.
Dicussed our plan of action for this week.

Wei: post the survey info. and send it to svn.
    Create project list idea.

Josh: Create status message.

Plan:
1. Send project.
2. Make a prototype by Wendesday.
3. Wednesday, network w/ students.



Wei: will post at 6 pm.

2015-03-01 17:54:44
```

END File: /Team Documentation/Meeting Notes/note20150301-Team-meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150302-Team-meeting.txt

---

2015-03-02 14:17:14

Josh: Maybe we choose

Wei: we could have round robin rotation.

Wei: we want to prototype to be more hands on. Touch and feel.

Josh: characteristics.

Josh: none of button should work.

Wei: when you click that button it would do this.

Josh: the UI expresses the visual interface and what it does for them.

Miguel: goal of prototype.

Wei: Conveyance.

Miguel: We could present them an idea w/o a prototype.

Wei: Goals: conveys what we want to do with project.
to get feedback

Josh: we outsourcing the decision.
We should have a list of questions and things we want to get out of their feedback.

Wei: maybe we should not do more than 3.

We are all splitting up to work on designs.
2015-03-02 14:39:47

2015-03-02 15:14:36

Thomas: likes the idea of codeback.

Wei: the only problem is the scope.

Thomas: we scope it down.

We set a framework of repository.
Instructors could load exercises and communicates with that.
A file system where you drop exercises.
We define how the exercises are built.
We have backend of how the exercises are built.

Wei: Instructors make JUnit tests and we make a small framework.

Thomas: define an XSD that professors stick.
Define a base framework.

```
JOsh: it should be simple enough so that someone who knows how to use
 a computer can use it.

Everybody like the idea.

Thomas: We have to scope it down.

  Professors can add tests.
  Testing our framework needs more than one test.

Thomas: scoping is our biggest risk.
  Before that our biggest risk was finding a project JP likes.
  We have full control over scope.
  Our previous risk we have no control over.

Josh: We are being more more direct.


Goals:
 Scope the project down.
 Integrating into bluej.
 Deck of cards.

Wei, Thomas: Deck of cards.


JOsh: what teachers see.

Thomas: rights and priviledges.
     do an exervicse.

Wei, Miguel: will work on the add aspect.

Thomas, Josh: add aspects

These are due Wednesday b4 class.

Thomas: shoot me an email with artifacts.
2015-03-02 15:49:36
```

```
END File: /Team Documentation/Meeting Notes/note20150302-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150304-Team-meeting.txt
```

```
2015-03-04 14:02:25


Wei: we create the exercises for now and pull from a local file.

Josh: I like to write code that can be scoped up later.

Thomas: so we keep our design moduler.
```

Wei: we create a get exercises class that we could scope up later.

2015-03-04 14:06:20

2015-03-04 14:51:46

Thomas: we focusing on two sections the do and add features.

Wei: we are just having a local repository.
We have do and integrate.

Wei: We might be doing a presentation about presentations.

Nate: We are doing a tiger presentation next Wednesday, 4th.
You could do a presentations on visions.

Josh: If you are going to do that then describe what you want to do and document.

2015-03-04 15:11:38

2015-03-04 15:20:17

Wei: we split it up to editors and users.
Editors create exercises for the users.
Users do exercises

As a user, I want to select exercises.
As a user, I want to work on a exercise.
As a user, I would to receive feedback.
As a user, I want to see examples of the exercises.
As a user, I want to see a description of the exercises.
As user, I want to see solutions.
As a user, I want to run my solutions.

Nate: we came up with elements.
Thomas: this project def lays out what we want to do and fits in an agile space.
List of stake holders.

Wei: Defined two different people who would use the program.
Users do the exercises.
Editor creates and edits the exercises.

Wei: As a user, I want know what exercises I have already completed.

Thomas: That b/c more complex.

Josh: the purpose of pairing is to create ideas. and meetings disapprove

Di: I think part of the stalemate: someone mentioned that we
do something and then go back. This felt.

Wei: Minimize risks.

Di: I did not know what was going and wanted to fall back.

```
  I like the idea of the create and invalidate cycle.

Nate: individualized work allows work and provides protection against
  shutting down.

Wei: Do you want to come up with architecture design.

Thomas: I will look into exercise formats.

   Can we define our first iteration.
   We should define an iteration.

Di: 1st iteration, click on the menu and launch the program.

Thomas: that might not take all six of us.


Thomas: Team meeting can address issues.

Agenda for saturday:

  -Touch base and update.
  -Form status update.
2015-03-04 15:49:38

2015-03-04 15:52:02


Josh: I will make presentation, table of contents, and take home message.
2015-03-04 15:52:23
```

```
END File: /Team Documentation/Meeting Notes/note20150304-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150307-Team-Meeting.txt
```

```
2015-03-07 14:01:59

Wei: Miguel and I programmed and uploaded the implementation.

Di: Architecture went well.

Nate: I think it went well.

Wei: I think we can just start.

Di: When the clock turns 2:05 PM.

Agenda:
 - Update ea other on the progress.
 - Status report creation and main speaker.

Wei: If we are moving clockwise then Miguel will be next.

Josh: Miguel is that Okay?
```

```
Miguel: That is fine.

Assign seating: Thomas -> Miguel -> Josh -> Nate -> Di -> Wei

Josh: We might need to add the scope to project definition.

Nate: who would like to go first.

Wei: We can go first.
  What we did:
  I did the writing.
  He did the navigation.
  We call it coding practice.
  Created a window w/ an editable text area and description.
  We did not get into designs.

Josh's Observations: 5 classes, window pops up, and displays
  pop up when you click run.

Wei: It just pretends to compile.

Miguel: I used JavaFX. I can't get it to run w/ BlueJ. It might have something
  to do with the jar file.

Todo: Make JavaFX work with BlueJ.

Miguel: The one we make works but if I change it then it stops working.
  BlueJ does not start.

Josh: Did you take a look at the debugging folder.

Wei: If we can JavaFX to work the with BlueJ then that would be awesome.
  There should not be a compatibility.

Miguel: The benefit is it is easy to code.

Josh: Have you guys thought about loose coupling and cohesion.

Wei: I thik miguel took care of that.

   We will replace the GUI with JavaFX.

Di: we did the design on the whiteboard.
  Today is not the best day for checking design decisions.

Nate: there is alot of unsaid things here.
  there is three designs.
  The firs is the architecutre.
  2nd is the UI
  The last one in the bottom corner is the files.
  Very similar to the Model-View approach.
  The saver and loader is for later.
  The GUI will only to talk to the back end via the interface.

Di: I will talk about the file formats.
  Three files:
  test file, exercise, and user state.
```

Nate and I decided to make file to store things.

User state files stores information about the user
like completed exercises, user info, current exercise, drafted code,
current test
Exercise contains the information about the exercise.
Contains exercise text, and the JUnit file that would test the user code.
Test file -- groups exercises togethers, and description.
Extra organization.

Any questions?

Wei: For the user state, when you talk about user info would that be a name.
For now we don't have to worry about user state.
So its out scope.

Josh: I think its important to note that the architecture
covers everything including many things that are out of scope.

Nate: The architecture will make sure the design will not break.

Nate: the user interface:
The UI lists elements.
The exact design does not have to match.
The results of the current exercise.
The submit button will run the test and display the results.
It could have a autosave.
Load exercise can help pick the test questions.

In the future, we could map the user stories through the diagram.

Wei: for the layout, we should keep coding bat in mind.

Josh: UI design implementers do not have to conform to exact layout.

Wei: Recommends adding a note to the image saying that the

UI design does not regulate the design implementation.

Wei: I really like the presentation on presentations.

Wei: Lets pick the top two
I like the presentation on presentations.

Di: Likes presentation on presentation.
Likes creating a vision.

Nate: Likes presentation on presentation, Group manuevers, vision.

Miguel: It might be risky because we might need to be experts in the subjects.
So its a credibility problem.
Our presentation could also not be good too.

Miguel: I like the interface/teamwork and design patterns.

Josh: I like Continuous Integration and Design Patterns.

```
Wei: We should think think about the message.
   The message I want to get accrossed in presntations always have a message.
   Why you should use design patterns.

Miguel: agree w/ Wei's message for design patterns.
   does not have a message for the other one.

Josh: agree w/ Wei's message.
  Continous I. would be how the CI helps the software manufacturing process.

Di: A project needs a vision b/c a represents the intent of team members
  and a road map for the project's future.
  Agree w/ Wei's message.

Nate: If we did the presentation it would have to be really good.
  You need a vision to hold people accountable and prevent deviations.

  Group manuevers -- Find the manuever that works for your team.

Josh: Should we talk about anything else?

Di: Will we talk about what we will talk about on Monday.
 Status update time.


Status Update
 - Wednesday, we got our members back and oriented them.
   --Splitting into pairs allowed Di and Nate to regain familarity w/ the group
    and current process.
 - Wednesday
    --Created a project definition.
    --Created a list of entitities.
    --Created users stories.
 - Wednesday, by splitting up we accomplished two things.
 - Over the weekend.
    --Created an architecture and a UI design.
    * drafted
    --Finished attaching an extension to BlueJ that pops up a GUI
    --Created a list of 6 presentation ideas.
    --We each picked our favorites and discussed messages.

    --Organized our status update and elected speaker.
      ---so that we are ready to go.
    --Created a system for decided who is the speaker.

Josh: We just need to discuss what we will do on monday
  I propose we do the release and iteration planner.

Di: that would be nice. We can say we are doing an iteration.

Wei: Tracking issues.

Wei: I will look into JavaFX tomorrow.
  Report on monday.

Josh: We can pick a presentation choose.

Nate: It would be nice to go over the design on monday.
```

```
Di: The design will help us plan the release.

Wei: we should talk about that first.

Sent a meeting summary email.
2015-03-07 15:22:32
```

```
END File: /Team Documentation/Meeting Notes/note20150307-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150309-Team-meeting.txt
```

```
2015-03-09 14:09:12

Thomas: did the research.
  XML would be the easiest.
  Java's built in DOM and Sax parsing.
  Java can write XML.
Thomas: Using the Document building factory.
   in Javax.xml.parsers
2015-03-09 14:28:58

2015-03-09 15:38:29


TOC

Thomas: will write a readme.

  Inside the project there will be a Git folder.

Today summary:
  -We got notes about JP to make it in a state that it is runnable.

Decisions:
  -Restructure repository so that its readable.

Delegation:
  -Wei is working on the top level readme.
  -Thomas is restructuring.


Todo:
  -Project needs to be runnable.
  -Build file needs to be complete.
  -Wiki.
  -Internal documentation.


Accomplishments: Wei, Thomas, Miguel made XSD and initial XML
  for file formats.
  To test the parser.

Delegation:
```

```
  -Miguel will do internal documentation
  -Di will do the wiki.
  -Josh will do the build.

2015-03-09 15:48:29
```

```
END File: /Team Documentation/Meeting Notes/note20150309-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150311-Team-meeting.txt
```

```
2015-03-11 14:56:52

DI: Discussion went the way it did b/c the fear to appease JP.
  Its a balancing act to keep him in the loop
  Whatever we do we back it up.
  We talked about a process model to fit what we have done.

Josh: Why do we need the iterative model?

Di: Lets go back to build the extension

Josh: Maybe we failed to educate him on the process.

Wei: We were on the spot.

Wei: did we actually pick two process models

Di: we did. Our project picked iterative.

Di: We would pick one process model and stick w/ through the rest of the
semester.

Thomas: I have been w/ companies that switch process models.

   His expectations are for us to pick a model and stick to it.

Di: We picked XP on the spot b/c we are afraid of meeting your expectations.
  Now that we think about it, we feel like this fits better.

THomas: what fits our time better?

Josh: Loosely define.

Di: Our project does not have to be completely done.

JP: Incremental and iterative is a good idea. That is what you have been doing.
  Get a little

Josh: we expected that the the end of each of week we would have a product.

JP: Iterative you repeat the same steps.
    Incremental each previous peice is done.
    Using same process.
```

```
   What is an iteration?

Di: a set of features.

Josh: so we just promise something and deliver what we promise in time.

JP: The time is always an estimate as is the quality and the quantity.
  What if you come up short and what if you come up late?
  As soon as you get in the danger zone of missing that time
  then you let me know.
  Managing expectations.

SO then how you would manage the requirements
 How you will make those decisions and techniques is the model.
 You could the techniques of say XP.

  Choosing a named is useful for communications.
  But a model might be the right fit so you can say how you are modifying
  those.

Di: As long as we are telling you our characteristics.

JP: You know best who you are and what you do.


Thomas: is on file chooser.
Nate: Compiler.
Di: Process model.
Josh: Process model.
   Presentations

Josh and Di are meeting at 1:45 pm
2015-03-11 15:54:35
```

```
END File: /Team Documentation/Meeting Notes/note20150311-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150312-Team-meeting.txt
```

```
2015-03-12 14:17:03

Di and Josh in a meeting.

So incremental is about dividing the work.

Di: User stories help us define our increment?
  What do we do and how long do we do it
  After a increment, we go back to our user stories and repick.

Josh: We are already kind of incremental.

Di: Our design allows us to work on different classes.

Josh: How will we communicate internally and how often?
```

Di: We have three meetings a week. MW in class and Saturday at 2 PM.

Josh: When will we communicate with customer?

Di: Right now we communicate once a week.

Wei: (talked to JP) He likes the work we are doing, but,
does not feel like he is in the loop.

Josh: We know events that we will communicate.

Di: We want to communicate when set an increment?
We communicate with them at the beginning of the project.

Josh: Do we need the customer to verify.

our user stories?

Di: I think that would be great.

Di: we can priotize and they can verify user stories.

Di: Do we need to consider how increments fit together?

Josh: when we try to fit the incremental peices together

Di: Should we do it ffrequently?

Di: Each time create something significant the customer verifies it.
Tell each person that they can send their completed work to the customer.

Josh: What do we do if User stories change.

Di: A user story has not been approved and the customer does not want it.
We have a user story in development and the customer does not want it.
--Stop and shift focus to the new stories.
--Fail safe: our user stories our not final.

Josh: It sounds like each artifact including user stories are seperate
artifacts.

Di: This happens in someone's head automatically.

For the same of simplicity, we can just make an arrow.

Josh: how will we maintain coding styles?

Di: Not sure that fits in the process model?

Backlog: Coding styles

Di: Put it after testing and before delivery.
Almost like a coding review.

Backlog: Integration
Backlog: Artifact Iterations.

Josh: How will ensure that the product may be built from source?

Di: we can put that in our testing phase?

  This occurs to me that we need to internally verify a component.

Josh: Each completed work needs to be tested thoroughly.

DI: for a parse, we need make sure it compiles and works.
  Then does it integrate with other components.
  Then does it adhere to our styling.

Josh: Documentation.

Backlog: documentation.

Josh: How do we plan the entire project.

Thought: How do plan the increments?

Di: I like the release planning game.

   Its nice b/c we can give them the plan and justify what we are doing
   and set the customer expectations.

   You can add them to the release planning?

Josh: How do we design increments?

Di: Two ways: devide it by users stories.
    Divide by design.

   Advantages of the design is priorities.
   Advantages is the design is that it is easy to peice together.

   We could take the user tories and figure out what classes align.
   For example, I want to see theresults of my compilation
  so you would have to work on the GUI and testsers.

  Divide into classes for user stories.
  and divide up based on those class that cover users stories.

Di: An iterative process is repeated.
  Incremented is divided

Di: We check our status as compared to our release plan.

Josh: So might have to readjust the iteration plan.

Josh: So if an increment is a set user stories
      then what is an iteration?
2015-03-12 15:19:04

2015-03-12 15:22:40

Josh: So an iteration seeks to make progress towards a user story.

2015-03-12 15:45:03

END File: /Team Documentation/Meeting Notes/note20150312-Team-meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150314-Team-meeting.txt

---

2015-03-14 14:16:47

BEGIN WEI'S NOTES

Agenda:

Each person reports on the progress they made.
Discuss the process model.
If the process model is ready then we can make a final decision to use the model.
Choose a specific presentation topic.
(As time permits) Plan the upcoming week.
Plan the status update message.

Miguel worked on the GUI. Currently fixing a window's compatibility issue.
Created the list area for exercises.

Nate created basic junit runner, 2 design documents proposals (detailed design of junit runner and Upper level design),
researched compilers in general, and researched how to implement the compiler into a bluej extension.

Thomas got a working xml parser created.

Wei talked to the professors and found out more about individual portfolios, improvements to interpersonal communication, and tying him in more.

Josh and Di worked on the process model together. Created a custom tailored process model for us.
Pulled elements from XP, incremental , and iterative models. Multiple increments for pairs to work on.
Each has an interaction as well. Might include release planning phase from XP later on. Process model is still in draft phase.
Iteration planning includes dividing user stories, prioritizing those tasks, estimates for time for those tasks, select subset of tasks to work on.
iterations after that. Currently missing characteristics in the process model draft at the moment. Process model gives an idea of what group should do.
Di wishes to clean up the model at a later time. Draft right now so it's malleable.

Biggest concern to process models is that it'll leech off our progress instead of help it.

Process model should serve us and not the other way around.

Question: does this affect our overhead?

Nate: if we can run through it and have no issues, then it's a good process model to send in.
Josh: we could just tell jody it's a draft.

Di: Need to spend time as a group to plan out releases and go through the model formally to check for overhead.

Wei: In the short run it'll slow us down, but in the long run it'll help us out.

Thomas: Try it out before making a decision.

Miguel: agrees with Thomas

Tried out process model

Everything needs internal and external verification.

Worked on User story complexity metrics spreadsheet and User Stories.csv

Need a way to synchronize google drive with our SVN.

A way to address user stories dependence with one another.

END WEI'S NOTES

Thomas: Has a working XML parser.
   Just two calls everything else is done internally.
   Easy to integrate and move forward.

Wei: Worked on the repository and the readme's
   We are missing an extension an extension jar.
   Missing ctxt.

Thomas: Class files should not be in repositories.

Josh: Typically we don't include products of the build in the repo.
    The jar file is in +libs folder.

Wei: Show JP the good stuff and bad stuff.
   I talked to him about the individual portfolios.
   Anything that we do could be an artifact including a pic of us at the pub.
   Annotate: this was a team builder and this is why it is important.
     What this is and how it was useful.
   I tested miguel's build and could not get it to work on windows.

Wei: JP said there is no way you could switch projects.
   He expects a result.
   Told him about how everybody is working on something.
   He said wow we are doing a lot of stuff.
   He was happy overall.
   Our biggest issue is communication
   We have hard time communicating to each other.
   Getting ideas to each other's head.
   It sounds like we are jumping to conclusions.
   When I make a decision, then I forget what went processes went into it.
   Conveying a decision w/ more backup.
   I feel like we have a bunch of good ideas well enough.

Josh: If people conveyed their ideas w/ support then would that solve the problem.

Wei: It might create another problem.
I have been thinking about metrics.
That will probably be his next question.
How people see those message.
It takes us about a week to make 3 decisions.
Before we were having trouble with this.
Do the math, it will take us years.

If we do research for every decision then it will take us too long.

Josh: At least show the reasoning why.

Di: our next item is to discuss the process model.

We pulled elements from incremental and iterative process model.

Josh: Incremental means that the project is divided into slice.

Wei: there will be multiple

Josh: Are there any questions?

Di: It gives us an idea what we are going to do and how long we are going to do this for.

Are there any questions?

Wei: Tangent: Stuff in process model is mainly for others outside to know
An outsider could be told our status.

Josh: Its a facade of organization.

WeI: Other expect us to do something a stick to it.
If user stories get cut off then people know why.

Josh: This process model could be improved over time.

Di: event though our process model represents us well.
If we find that it constrains us then we change us.

Nate: I don't want it to be a leech.

Miguel: This was pretty much what I was thinking.

Thomas: Process model sounds good to me.
Agrees with Nate.

Di: It should serve us instead of us serving it.

Nate: What is design map?

Josh: Its the one that a Nate and Di came up.

We could present this to JP as draft.

The question is how does this process model affect our overhead.

Nate: We are kind of already in the process model already.

Wei: I think we are closer.
We have not done any of the planning.

Nate: its load and run that we would be working with.

Wei: If we can run through then its a good process model.

Di: I am looking at additional things we would need to do.
We would need to do the release planning game.
We would need to time for that and mapping out increments.
We have informally picked users stories.
We are kind of all over the place in this process model.
We have created the prototypes.

Josh: We could just say that the priorities are done and we need to do
the estimated time.
and verify with the customer.

Wei: In the long run this would be good, but, it would take additional time
to orientate.
We can show JP how long it takes to do each thing.
It helps us plan time constraints.
Slow us down in the short run.

Thomas: We can theorize but not actually.

Miguel: Agrees w/ Nate.

Josh: Would you like to use it for.

Di: Can you define that.

Josh: Nate, what does that mean?

Nate: Try it out.

Di: so this means that we would need to estimate user stories and verify
that with customer on monday.

Josh: Should we run through the agenda.

Wei: Yes.

Nate: Yes.

Josh: The plan
- send the process model draft to JP.
- Monday we do the planning.

Wei: Status Update.
- Josh would be the speaker.
- We created a draft for the process model.
- Miguel created the list area for the exercises in the GUI.

```
    --Has an
  - Nate created JUnit runner.
    --Researched compilers and integration.
  - Thomas researched XML parsing.
  - Wei talked to professor about inv portfolios and interpersonal
communication.
  - Wei learned about how tabulation can affect views on different computers.
  - Di, Josh worked on the process model drafts.
  - As a team, we learned about team process model.
  - Nate made proposal design documents.
     -i.e. a detailed design of the JUnit runner and an upper level design
      of the project w/ conenctions betweeen components.
  - Thomas created a demo or the XML parser.
    --prototype.
  - Wei created an the root readme draft.

Di: Lets call a break.
   3:30?

2015-03-14 15:17:44
2015-03-14 15:40:26

Josh: WE will send an email to JP saying this is draft process model.

Nate: Running through exposes errors and removes ambiguity.

Wei: Finished the readme and will close the readme. It will be committed.

Josh: In the release planning, we estimate th time per story and justify it.

Di: Check out Nate's design.

Di: Should we map the components to the user story.
   Tally up components for users stories.

Josh: So user story of selecting exercises gets a score of 5.

Josh: Internal verification means that another member of the team verifies.
      External verification means JP verifies.

Di: Evaluate each user stories invidually w/o the context of the other story.

Josh: For each one lets think about how we can test these.

Josh: We should have parrellel SVN and Google Drive folders.

Nate: I am fine with that.

Josh: Confused about As a user, I want to see examples of the exercises.

Josh: I get an error javafx.Application not found.
  So apparently the JavaFX is not in the +libs.

Josh: The purpose of this activity was to give our test.

Wei: The exercise loaders have already

Nate: is this activity helping us.
```

Wei: Yes, because we can see how these stories move together
and dependencies.

Miguel: I think this is helpful. We can see what needs to be done.

Josh: Our process model needs a way to address user stories dependencies.
There is no documentation specifying dependencies.

Wei: I have dependecies in the Justification.

Josh: It might be nice to give each user story an ID.

Wei: The highest prioty is one b/c it is the largest and other user stories
depend on and the ID is #1.
Its the most independent user story.

Miguel: Agrees w/ Wei.

Nate: A good way to priotize would be to walk up a user scenario.

Thus, ID #1 would be first.

Josh: Does our process model tell us what comes next.

Nate: will we do more than user story at a time.

Josh: Yes. b/c of incrementation.

Wei: All of them are based off #1.
#1 is independent.
#2 is based on #1
#3 is based on #1,6

Nate: I think everything is dependent on #2 but #1.
and #2 depends on #1

Josh: so #6 is priority b/c users want feedback.

Wei: #4,5 are like the icing and they may be done towards the end.
#3 is stuff on the GUI.
#1 is working inside the GUI.

Nate: Priotization has been based on user preference.
Alt prioritation would be risk and time.

Nate: so we did release planning minus customer verification.
It took a little long b/c of confusion
but those may be ironed out.

Wei: Orientation to the process model can slow us down.

Josh: If we did send verify the result of the release planning
then we would send him a list of user stories w/ their estimated times.
in a sorted order. We would tell him what subset we would be working on.

Nate: so on Monday, we would be testing Increment planning.

```
Wei: say that we have a draft and we will see if it works in our favor.

Nate: I would like to try and use that with current project.

Josh: in the mean time we could use tasks to organize what we will do.

Josh: We would have to manage JP's expectations if our process models.
     changes.

Wei: we should we reiterate that we are making progress in other areas
  as well.


Nate: It would be nice to have aproduct by spring break.

In this meeting we tested th release planning phase of our formal process model.
2015-03-14 17:09:00
```

```
END File: /Team Documentation/Meeting Notes/note20150314-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150316-Team-meeting.txt
```

```
2015-03-16 14:34:30

JP: How are you tagging releases?

Josh: Typically there is a trunk?
  We never discussed tagging.

JP: You seem to make decisions w/o considering implications.
  Think more about who the users were.
  In agile, tools you need to be looked into further.


  Liked the purpose.
  I get you are not trying to hold us to this.
  Its unclear that this defines what increments means.
  Okay with that b/c of definitino
  Unsure when we get to the process model.
  What is the out come of project planning?

Di, Wei: User stories, Vision, and Definition.

JP: Project planning: you define the stories once.

Di: According to the document it does look like we do the user stories once.
   But we wanted to add and change the

JP: The whole thing is linear. Where is XP is a circle.
   How do we know that this matches?
   Consider what you are doing?
   What are you doing that is different than CS1 students that demonstrates
    knowledge of engineering?
```

```
Josh: So you are looking at the disconnect b/t our custom process model
    and what you are actually doing?

JP: Start w/ something that you believe in.
   Find attributes that you can use.
    Its too grand.
   Just do things that demonstrate best practices.
   You don't have time to invent new stuff.
   Worth doing so we can get to this discussion.

Josh: So we can say we use test first development.

Di: SO yes. Design based on outcomes.
  Identify a small set things.
  Ensure you were adhering.

Josh: so I feel like this was education.

  The core is what we need to get where we wanted to go.

  We could split user stories into tasks.

2015-03-16 14:57:41

2015-03-16 15:10:23

Thomas: exercise object.
   (demoing)
   Easily integratable.

Nate: Shows that we may compile a string.
  Not sure if we could get stack trace.

Thought: need to determine insertion point.

JP: How is this going to integrate.

Nate: Problem: relies on the calling thread.
   so what if the user does an infinite loop.

JP: You might get resistance from educators who want students to see the
  whole thing.
   Highly entrenched
   They might write you off.
   I am thinking of even greater success.
   Your project as it lives on afterwards.
   Acknowledge that we did not have time to address this issue
   Now the people are happy b/c they are heard.
2015-03-16 15:49:02
```

```
END File: /Team Documentation/Meeting Notes/note20150316-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150318-Team-meeting.txt
```

```
2015-03-18 14:13:08

JP Talked to faculty about ext.
 Expected to hear of:
  - Coding bat.
     -- does a great job with method level problems with a self contained
        context.
  - Practice it.

Liked be able to not sign in.

Practice it had better problems.


Value to a bluej would be integration w/ IDE as intended.
  - Full class
  - Multiple class.
  - Imports
  - Javadoc.
  - Surrounded framework could be provided.


Using inner nest classes to do this.

STill doing the same problems but through the IDE


Wei: It was hard to guess how to integrate.
   Moved the run button.

Multiple listboxes in GUI vs Multiple questions.

 and also no GUI for the question.


Wei: Some of his colleges expectations might be out of scope.

Josh: What does it look like and what does it do?
  Maybe we could divide into groups and do presentations.
  and when we get back we could change.

Josh: I got two close to be done.

Josh: Propose.
  Our project could produce the class files with the problems.
  and a dialog showing the results of a test.

  What will you do and what do you need.

Wei: needs the parser code and compiler code.
   Thomas needs to upload his code by Sat meeting.
   Nate needs eventually upload his code eventually.
   Something in the middle that loads up everything in a folder.

   We will have a folder with exercises that we will be loading files.
    We might also have this in the repository.
    So it might need a link and a subfolder.
```

```
  Three people on the

Thomas: Lets do one Wednesday next week.

Saturday's agenda:
 - Roundtable report.
 - Presentation practice.


Decisions:
 - Committed to making a product that produces the source code and
   checks the student's owkr.

Delegation:
 - Di, Josh will be working on the presentation.
 - Nate, Thomas, Miguel, and Wei will do coding.


JP Comments:
  It might be okay if were navigational or code editting.
  Users learning a new interface could take away from blueJ.

We present on Wednesday.
2015-03-18 15:53:41
```

```
END File: /Team Documentation/Meeting Notes/note20150318-Team-meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150321-Team-Meeting.txt
```

```
2015-03-21 14:07:10

Agenda:
 - Everybody reports.
 - Presentation.


Wei: Created an exercise class.
 description, title.

 Everyone of the exercises is an exercise object.
 Its easier.

 The controls show description and examples when the user selects.

 The right click does not show for every object.
 Right now it only shows up in exercise.
 For the future we may use packages to control the right click menu.

 Miguel created the execute and execute test.

   and exercise description.

 The execute test shows the test dialog.
```

```
 Its semi connected and we might have to use an lisener.

 Created the state manager.
  Miguel thought of and I implemented.
 Holds the state of the entire extension.
  If the guis have been initialized.

Di: Will execute run the test automatically?

Wei: I believe so.

Di: so the run button will rerun the test.

Di: so name it to "rerun"

Wei: that is a good idea.

Di: I see you all are using scene builder.

Wei: We have not implemented the compiler.

  The VM must be reset when we run it in BlueJ
  We believe that it is a problem wiht JavaFX.


Josh: What type of process model would you say this maps to?

Wei: Ideas and implementation.

Josh: Code and fix it.

Wei: there is also the prototyping we are doing.
  Design -> implement -> Demonstration

  Add pair programming, designs, small iterations.

Di: I have a link to the process model.

Josh: at this time, Wei and Miguel have went. Does that leave Nate?

Nate: I did not work on the GUI.
   Started to dev the v2 of JUnit runner.
   Concern: runner does not work in an infinite loop.
   I made a new process to run the test.
   The listener will notify the GUI when its completed.
   Use a socket to communicate b/t processes.

Josh: Was does not threading solve this problem?

Wei: the stop thread is deprecated and a security risk.
   This way you can stop bad code without killing BlueJ itself.
   This will be a lot of stuff to test.


Josh: How will this effect the maintainence overhead?

Nate: I will be trying to get the Maintainability lower
   For example, getting the BlueJ library folders.
```

```
   Version 2 only works with methods.

Josh: Will everything else be uploaded?

Nate: I will upload the latest version.

Wei: I will upload my state class.

   We did not test it in different OSs.
   I just uploaded.

Nate: I just uploaded as well.


Josh:

Di: I would like to go over the structure.
   Similar to the Tiger Team presentation.
   Encourage the audience to use design patterns.
   First what it is and not.
   Next Types.
   Next specific types.
   We are not sure if this message is strong enough.

Wei: The message is not focused b/c of the three points.

Note: this 2nd time that thomas has not been at meeting
  and informed us.


Di: Work on the presentation and make sure it is in the a good state
  for our practice next time.

Di: Next Sat we can divy up parts.

Josh: Are we choosing this one.

Wei: I am okay with that.

Miguel: I am okay with that.

Nate: As long as we know what we are talking about.

Wei: I might do some reasoning.

   We used the Singleton in the State class.

   Its a global controler.
   A singleton is like th God class.


Wei: Want to persuade them to use a design pattern.

Di: When do want to think about design patterns.


Di: So we will meeting on Wednesday at 2 PM and discuss our message
 and research findings.
```

```
Di: Wednesday we can discuss our progress towards our designs.

Josh: On Wednesday, we can make the email and
  make sure that he has expecatations from the design.


Di: Iron triangle: scope, cost, and time
  Uses fidelity as well.
SOURCE:http://availagility.co.uk/2009/12/22/fidelity-the-lost-dimension-of-the-i
ron-triangle/

Di: Fidelity is quaility of the project.
  We were just developing things to get it to work.

Nate: the GUI is produced just to show the windows.
   Take a big block and refined like a sculpter.

Josh: Its like we are sculpting the project.
  I will be moving.
2015-03-21 15:37:36
```

```
END File: /Team Documentation/Meeting Notes/note20150321-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150325-Team-Meeting.txt
```

```
2015-03-25 14:05:37

Miguel: Looked an example on design patterns.

SOURCE: http://www.oodesign.com/

---
Miguel: Use a specific design pattern.

Wei: These should convey our message.

Nate: I found a good website.


Wei: A design pattern has consequences.
  Has parts: the name, the problem -- when, elements and correlation,
    consequence

Possible messages:
  - Simplify complexity.
  - Choose the positive trade off.

Wei: You cannot use an object oriented design pattern for a functional language.
Di: You want to use a design patterns to solve reoccuring problems.

Wei: Use design patterns in the design phase instead of the coding phase.
  Design patterns should make coding easier.
  Important part of implying design patterns is knowing your problems.
```

```
  The more time we spend with design patterns the easier it is code.

Nate: They are like math formulas.
  It will reduce work and simplify communication.

Know your problem before choosing a design pattern. - More problem analysis.
Design patterns can reduce work later if picked correctly. - Would involve
statistics.

Message:
Know your problem before choosing a design pattern.

Agenda:
 - Presentation.
 - Demonstrate the project.
 - Create status message.
2015-03-25 15:28:12
```

```
END File: /Team Documentation/Meeting Notes/note20150325-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150404-Team-Meeting.txt
```

```
2015-04-04 14:03

Agenda:
- Discuss backend design

Josh
I feel like we ve made good progress.

W
You guys just checking out the new Junit runner?
(yes)
J
Noticed minor exemptions, but a result of build.
What we need to do is make an easy build system.
General design of classes could be
For example, some things hardcoded in state manager.

N
Tried swapping them with BlueJ calls already.

J
We proved that it works, let s make everything more abstract.
Refactoring is our next step.
Lots of magic numbers.
Make clean, beatuifl, error free code. Clean up the build.

W
Maybe apply checkstyle?
(sure)

J
Apply design patterns too, maybe?
When it s loading exercises, should load from inside jar / local repo / etc.
```

D o n t implement that, but make design abstract enough that it can transition there.

W
Could use design patterns like wrappers for future implementations.

J
L e t s not rely on design patterns; l e t s just clean it up.

W
A lot of stuff has been hardcoded - it had to be demoed.
After demo we should clean this stuff up.
Should be able to refactor it.

J
Let me show you guys something (screenshared)
If I create a jar in bluej, and check i n c l u d e bluej project f i l e s and I make that jar..
It will actually take a while, because it will process a 15MB file.
But what I get is a jar file that includes lots of files.
When it finishes itll be 6MB
W e l l include everything like this .ctxt files, .bat files, .xml and even +libs folders.
So this means if t h e r e s a resource that you need access to, just put it in the jar and
load it internally.
Be aware of this feature & how we can use.

W
How do you want us to use it?

J
If t h e r e s a file the extension needs to refer to, include it in the project folder
and check i n c l u d e bluej project f i l e s to refer to it.
For example, resources folder can be included if copied into proj. dir., bluej will make
jar including that folder

N
Disadvantage, c a n t dynamically include more xml files

J
Need abstraction to make that work

W
We have prebuild exercises & external folder to create exercises?

J
No, I want to redesign and have it flexible enough to do both.
All included exercises live in jar.
And maybe have a way to change repos.

W
Probably w o n t get done tbh
Have a few pre built exercises & load from repository.
Should be able to do this with a redesign.

J
We re not implementing it.

W
Just refactoring to implement in the future.

Have an idea about exercise locations.
Create a preference document, like in games, that saves preferences like where files would
be loaded from.
It could keep records of previously completed exerices.
Out of scope but possible

J
Yes

N
BlueJ includes it.
setextensionpropertystring() - sets property of extension into bluej repository.

J
That was my findings. Anyone else want to report theirs

W
Thinking about what happens after load.
After putting load button, GUI pops up to load new exercise or new exercise
Lets them go back or refresh whole exercise.

J
Sounds cool

W
Might not happen right after load. Would be good as a tools option.
In same dropdown menu, as an option.
Might be out of scope, but follows load user story.
Exercise location: aside from being inside our jar, could be in a local directory.
In light of OSX/Windews differences have code check OS before creating dir

J

N
ExtensionPathGen

W
Create own dir. inside user/BlueJ wihch says exercises , loaded in there.
pull from that folder specifically, and load it. If not, create new one.
(liked)

J
Not a lot on my plate, but what about everyone else?

W
Algorithms test, psychology test, econ test all in this week. + This class.
Should be free after this monday.

D
Directing. Plan rehearsals mondays / fridays. Will have this on plate through

end of semester

N
Pretty free. Have a rigid bodies class, kind of easy. Test on monday.

J
Did you guys hear anything from others?

W
Miguel might have calc 3 test, can t say for sure.

N
Report on what I ve done:
main concern - project can load and execute exercises.

W
na load through xml but hows test on junit runner?

N
Not caching anymore.
Runner should work if setup crrectly. Not tied to loading exercises at all.
Has to be hoardcoded and in a specific spot.
That part of backend - takes xml and makes a spot of junit to see - needs to be created.
JUnit runner integrated, and design ideas proposed in the sandbox.
A little worried about the understanding of how this works is lacking; lack of bluej apis knowledge.

W
Tried looking for bluej accessors to junit

N
Runner uses BlueJ s Junit jar. No need for manifest changes.

J
Worried that members don t understand BlueJ API and how the code works?
(Yeah)

D
I can attest to that.

W
A lot of it has to be connected to javafx. Many classes are javafx handiling.

N
GUIs and FXML are all 1-1, and actions are one to one. Every GUI has execute action.

W
Do you know a better way to do that?

N
Might be able to bake them in teh calsses as privates stacti. Will still exist, but not a node.

J
Since everybody s busy what if the busy people shift to the presentation?
It could take a lot of work, but maybe if everybody has something to do,

let s put on a
presentation.
Less work for all those people; more free people to design.
Next presentation date undecided.

W
Can last presentation be a draft for the final?

J
That could be our EC presentation to get feedback.

W
We could do that.
Presentation on vision statement?

J
Vision Pres
Draft Pres

N
Time. 6 weeks?

D
4 weeks. Everything is due the 4th.

J
All of this month.

N
Worried about manhours. Presentations + notebooks + projects = lot of time.

J
Others might have time. I do. Nate does. Wei will. It s plausible. Shoot for vision pres
on 15th.
Then present final pres on 29th for feedback.

W
Regardless, we have to do the final. Doing it right before allows feedback.
Regardless of manhours & metrics, it still needs to be done.

J
Would be doing an extra presentation as extra credit.
How much to gain if we do this?

W
It would be good to practice.

D
Would this be approved?

J
What about last wednesdays pres.

N
I imagine final one to be about the final product.

J

I d like to talk to him too.

W
I can swing by his office and ask about that.

J
Also ask for final pres advice.

D
Is everyone passionate about vision pres?

W
Even a left field pres would be okay as long as we re passionate about it.

J
Our presentation should be something we relate to in a personal way.
ie here are some pointers.

D
Vision statements stems from our personal experience.

W
How about BlueJ extension presentation?
I can make a draft.

D
What s next?

J
I wanted to get a feel for why we weren t productive.
In our status we should say we were busy and didnt produce a lot of stuff.
So that s what we should do in our status.

D
what should our client expect

J
Our decision was to spend time redesigning.
There are some issues that we can cover.
That s our direction

N
not a design issue because it doesnt exist yet.
what happens fater load button pressed doesnt exist.
no class that does anything.

J
Do we clean it up or design a load or do both?

N
At the end of the semester, I really want to make it run and demo-able.

D
I want to do both, but is that possible given our time

W
Yeah, but it depends on our time investment. Must be cleaned before adding more stuff..

N
Can be modulized. Someone can work on it while someone else cleans it up.

W
Risky - cleaning things up neglects the knowldege of the module.
Need prior info of the module.

J
How much more code would it take to create this load feature module?

W
One or two classes.

N
It wouldn t be that much since compiler exists - would be part of that module
Main issue is knowing waht to do.
Knowing what the module ..
Wouldn t take effort if you knew what to do.

J
Needs some research.

N
One line of code = 5 hours, as an example possibly.

J
Proposal: 11th to get load button working.

N
I like deadlines. (not sarcastic)

W
Currently, load button just loads exercises. What else should it do.
Gui is making menu from xml files.
Used to be able to, from state manager, remember loaded exercises.

J
Miscommunication?

N
Loads exercise, does nothing with it. Doesn t create a new project.
Doesn t do anything that benefits the junit runner.

W
Seeing as how a lot of people are busy, the 11th is a good deadline.

J
If we finish early, clean it up.
The 11th is for a ready state.

W
I made a draft / outline of our project notebook.
For personal portfolio, hard copy needed? & Project notebook, pdf needed?

J
Another thing to ask Jody. Notebook or portfolio a physical copy?

```
W
11th deadline good for mentoring new people into code roles.
will make a list of concerns, to place on GDrive. Under team documentation
-> Things to consider.



J
Who wants to take circle notes on monday?

W
Two scribes? Notes can be compared and merged, and allows conversations.
Redundant. But like big brother.
Scribe and scriber s scribe (only notes scribe s speech).

N
Assignment of scribes as part of agenda.

MONDAYS AGENDA
Status update
ask miguel / thomas for questions
go over what needs to be done again
pair programming on pres / backend design on load button

End time: 2015-04-04 15:22

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Status Update

- M/T integrated parser.
- W reviewed pres with D/J
- N looked at caching issues and solved them. New JUNit runner works.
- Rehearsed, gave, and recieved feedback on presentation.
- Reflected
- Started mishmashing hats. Me taking notes.
- decided to redesign backend
- answer  where exercise files live  and  what happens after  load
clicked
- talked about future of project & class
- set internal deadline for load button : april 11th.
- talked about pres dates. 15th for new one; 29th for draft (if possible).
```

```
END File: /Team Documentation/Meeting Notes/note20150404-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150406-Team-Meeting.txt
```

```
2015-04-06 14:16

W
I read patch notes first. We should include them.

T
Use TTB - Trunk tags branchs
```

W
Take messages from log possible?

T
Manually

J
    I can do it

W
Doing this gives us a metric.
We should keep a stable version

T
TTB takes care of that.

J
So we re waiting for if you can get a draft presentation done as a presentation.

JP
Yes, then you re asking audience to watch presentation twice.
You can prepare part of it as a full presentation, and then compress it in the final.
But theres limited value in presenting a draft, except for you.
Question for you, aprchitecture wise:
I know you have a custom xml parser. Are you still doing that?

T
Yes

JP
My concern is (not from class perspective, but maintainence perspecitve) I have to maintain it.
I d prefer, even if it was overkill, to use library routines, even if they weren t as good.
That way I won t have to maintain it.
But I have no qualms from a class perspective

W
? Hard copy?

JP
Not necessary, but should be easily transferrable to a hard copy.
Navigable. Tabs maybe. Needs to be straight forward.

J
Difference between notebook / portfolio

JP
One is a track of personal contribution to learning, with reflections on contributions & learning associated with this course.
Other is a documented process for the team personally.
You also have the product which is fully documented.
What is you, and what is the team?
Artifacts can be shared between the three deliverables.

D

Wei will cover me.
(Wei)

N

W
I agree. Need to do more documentating, esp. if we split up the work.
We should also answer those two questions.

J
I think we can always document later.
Our deadline for end of week is load feature, & redesign and document.

N
Nothing funner than deleting a method.

(getting miguel & thomas up to date)

J
What if we could do two presentations. Take a piece of our pres in prep for our final pres.
Thomas, you

T
The only svn refactoring would be going to ttb.
Where the files go to? That s a minor issue compared to how they re deployed?

W
Inside the jar?

J
You can put anything in the jar and there s alos a way to read files in the jar.

W
Tests stored inside the jar can t be changed easily.

T
Exercise file shouldn t change after package.

J
Exercise should be duplicated from extension, so it can be reset.

W
This won t cause the caching issue, right?
(no)

T
Keep in jar, reference in jar, solves those problems.

W
Jar just needs to be installed.

J
Even easier than that it doesn t matter. You can do it eiter way.
We don t have to worry about that quite yet.
Start working on that and then redesign, etc.

W
Prioritize the deadline. Everything after that is cleanup.

J
This should be feasible. By Wednesday, we can get it done.

W
Can t get it done, I have tests.

J
Split into pairs.

W
Two pairs, one on each question. Come back and share what was found.

N
Two groups of 3? Or 2 groups of 2 and one group for pres

W
We can t do the draft thing anymore.

J
Still want to do the 15th.
People that are busy this week can work on pres.
Di and Wei on pres.
Josh & Nate on load button
Miguel & Thomas on exercise location.

W
Vision statement
How to make a BlueJ extension
Your audience and you?
Meet back at 2:50
Come back at 3.

(recess)

D

T
4 & 5 go together.
2 is extremely risk

N
Out of left field.

M
1 & 5

N
I recommend 5, it s fresher.

T
1 is narrower.

W
Team meetings I like more. We ve been doing them more often.

Vision statements are one time thing.
It did help us, but not fresh enough.

W
Much of it can be applied to the final presentation

T
Vision is more applicable.

M
Vision has hard time with content

W
    make to do list

JP
Clearly you re providing a build that builds outside of bluej
You re going to provide an outside xml file.. right?

W
We should try to work around 6

T
Refactor should solve

M
To change it you have to reinstall
You can check extension api and see if compatible with this version

W
Make popup that invalidates it.

T
Refactorable with 1.6
I don t know how familiar you are with annotating java.
It s just changing from using a scanner to putting annotations inside object
If you put an-, and unmartial, you can map them out.

W
Progress on two questions?

T
Starting to look into bluej preferences where we can read in uri or path.
TGoes towards grand scheme of ex. in public location

W
Already built in.
What happens after load?

N
Not yet.

M
Another issue.
Let s say they load an exercise, close BlueJ and open it.
exercise will still be there, but bluej won t know that it s an exercise.

When you load, they will createa new project.

W
If we re able to, we can call a class with specific clas sname.
If we call by class name

M
It would save

W
Best option to clear exercise?

M
Best option is to save it somewhere and save the path.
It d be annoying to keep pointing to that location.

T
We could also define constraints, make it clear that when you close bluej,
it s cleaned.

N
terminate feature delets the project, solves that problem.

M
What if it s a big exercise? Can t reload it.

T
Given the amount of time, we could just clear it.
We could go back later and have someone save it.

W
It s out of our scope.

M
Still good to know.

W
Create a button that says this is an exercise , which restarts the process
and makes it become an exercise button. This is out of scope.

Decision:
Document that the problem exists, and is in the backlog.

W
As a reflection, how did you like split and return

T
I like it, lets us tackle more issues together.

W
Lots of decisions made today.
1. Pres topic
2. Exercise state lost bug
3. Exercise location defined by user.

J
I am working on trying to get extension to compile for 1.6. Fulfilled.

```
N
When do we refactor? After these features are done.

T
I estimate a half hour for the parser. Could take me longer because I  m  used
to martialling first.

W
Whole thing could take up to monday 13th.

J
What are we doing between now and wednesday?

W
DW doing pres
T M doing prefs pane.
NJ doing load button.

T
Who wants to learn annotations?
I can refactor parser in my sandbox.

Decision:
Load button will follow Nates diagrams in his sandbox.

    Wei & Di come up with a message.

Wednesday s Agenda
- review  nate  s  load button
- progress
- report to stakeholder about progress and what  w e  r e  expecting

2015-04-06 15:48

Di's talk:
So the thing we have to cover today is the 2 questions:
Where do the exercise live?
What happens after clicking the Load button?

Lots of documenation work to do.
Personally don't feel like it's there yet.

Acheive those 2 questions then doing documentation, portfollio, project
notebook.

presentation ideas:

1. making a vision statement.
2. presentation on presentations. High risk High reward
3. knowing your target demographics. Knowing stakeholders
5. defining goals and success.
5. Team meetings
```

```
END File: /Team Documentation/Meeting Notes/note20150406-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150408-Team-Meeting.txt
```

2015-04-08 14:06

DECISIONS
- We decided to refactor the extension so it will work with Java 1.6, regardless of whether the school s machine images were updated.
- The URI will not be implemented; it s out of scope.
- A new window will open that contains the exercise description.
- XMLScanner class will be physically changed.
- Exercise class will have additional methods defined.
- Title of the exercise name will be the project name.
- Open source Creative Commons license included:
        - redistribution / remixing okay, but credit must be given and noncommercial only

ACCOMPLISHMENTS
Nate demonstrated his prototype:
- When Launch exercise is clicked, a new project / window is opened.
- The project is created into a temp folder, that is deleted on reopen.
- Its contents is cleared every time BlueJ reopens, or an exercise is loaded.
- Makes multiple temp folders so multiple projects are allowed to be opened.
- BlueJ s compiler is used to compile the test class.
Exercise location setter is implemented and completed.
Josh successfully built the project in Windows using a .bat file
We started reviewing our best and worst team moments.

FURTHER QUESTIONS
Where is the exercise description going to be?

TO DO
Everyone should walk through the whole extension after its refactored
Move the CCLicense from Google Drive to SVN
Continue to reflect on past team meeting experiences.

RAW NOTES
Need to share load button design (Nate s sandbox.)

Start implementing features & refactoring.

Thomas decided with Dr. Paul s suggesting: a UI section twhere a user can specify a local storage space or a URI for a remote storage space.

J will continue working on the build. He created a .bat file.

We need to understand the parser and the GUI

How team meetings can unify a team; use team meetings to unify a team.

What license should our extension be under?

If the intent is to be open source, a license is needed in the product.
You can say this is in the public domain.
Should appear in high level documentation and in the source code.
Doesn t need to be everywhere, but needs to persist.
Creative commons can work. It will recommend the correct wording / restrictions / etc.
BlueJ wants to put restrictions because they intend to control what they re releasing.

```
What were our best moments in general?
Our vision
Our teamname
Our first design iteration on the smartboard
 T h o m a s  s  jokes
When Wei bought drinks for everyone
Nate shows his stuff
Showing up Git and Chirp to each other
Getting members back

What were our worst moments in general?
Two weeks about process models
Deciding our project
Losing two members
Chirp / Git reuniting
When we started using Google Hangouts instead of Skype
Using Google Drive when we had SVN
Second presentation

Starting key messages:
```

```
END File: /Team Documentation/Meeting Notes/note20150408-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150411-1511-Name-Meeting.txt
```

```
2015-04-11 15:12
PROJECT NAME MEETING!
2015-04-11 16:10

DECISIONS
Birdfeedr is NOT the project name
Send name BlueJ TA to Jody / team

Title: BlueJ TA

BLURB
  B l u e J  T e a c h e r  s  A s s i s t a n t   supports beginning CS students and
instructors by integrating practice problems with automated feedback into the
BlueJ Java IDE.

SUGGESTIONS
BlueJ_CodingPractice
1. Generic
2. Underscore - hard to type when searching for extension
3.

BlueJ Coding Practice

Practice Problems via BlueJ: y o u  r e  going to practice  w h a t  s  available
1. It  d o e s n  t provide enough meaning to the audience
2.  D o e s n  t sound like an extension
3. Sounds static

Integrated Practice Problems Extension
```

```
1. Makes it looks like it comes with everything
2. Integrated into what?

BlueJ Integrated Practice Problems

Practice Problems with Feedback

Tutor / Reviewer / tester / educator / trainer / mentor / checker / drill
Analysis /

Programming Practices Extension

Code T.A.

CodeReviewr

Exercise Reviewr
1. What kind of exercises?
2. Exercise is ambiguous
3.

Code and Test

BlueJ T.A.
1. Implies it s only for teachers
2. Implies learning the BlueJ IDE
3.

BlueJTA
BlueJ TA ODD
BlueJ T.A. EVEN
BlueJ Teacher s Assistant




Programming exercises extension
Coding practice via BlueJ extensions
CodingBird

BlueB
Code N Check
CodingJ
```

```
END File: /Team Documentation/Meeting Notes/note20150411-1511-Name-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150411-Team-Meeting.txt
```

```
MEETING MINUTES
2015-04-11 14:10

AGENDA
- Did we meet our deadline? (Implementing the two questions: exercise location
/ load button)
```

DECISIONS
    Wei will ask for examples of the team notebook.
- Exercises will be deployed through a initial set in a jar, with the ability to download exercises later.

ACCOMPLISHMENTS
- Thomas and Miguel implemented file location for exercises.
- Load button implemented
- Deadline has been met.
- Finished brainstorming best / worst moments

UNANSWERED QUESTIONS
What s our project s name?
How are we deploying our exercises? Inside our jar with BlueJ s export feature?

TO DO
- Create a project name
- Wei will make a closer for the status update
- Assign agenda item to debug exercise button
- Create a key message based off of our experiences
- Compose Monday status update
- Create agenda
- Assign note taker
- Create a blurb

RAW NOTES

Exercises could be placed in a temp folder / in the jar.
Deploying it separately looks clumsy / unprofessional.
Create an installer? There are tools available
Prebuilt exercises + additional folders?

Troubleshooting compile error (revision 213):
Working as of Thursday night.
Compiler error. Duplicate classes were deleted.
Works after troubleshooting: duplicate classes with no source code.

Options:
- Deployed in jar.
- Deployed separately in their own zip.
- Starter set in jar, additional available for download.

It would be good to make sure the jar exercises are tested & add whatever code is
needed to make it work.

Execute button doesn t work. Exercise not launched.

What were our best moments in general?
Our vision
Our teamname
Our first design iteration on the smartboard
Thomas s jokes
When Wei bought drinks for everyone
Nate shows his stuff
Showing up Git and Chirp to each other

```
Getting members back

What were our worst moments in general?
Two weeks about process models
Deciding our project
Losing two members
Chirp / Git reuniting
When we started using Google Hangouts instead of Skype
Using Google Drive when we had SVN
Second presentation

Starting key messages / main ideas:
- Morale & how you can improve it (losing / gaining members; exploration /
unification)
- Conflicting / Unanimous Decisions: Conflicting decisions split morale;
unanimous the opposite
- Lingering decisions affects morale.
  - Process model
    - Wasn t considered until asked about
    - Agreed on a concrete process model.
    - Researched process model
  - Chirp / Git reuniting
  - Where to put what in Google Drive before refactoring
  - Final project decision (CodingBat)

- Identify decision making points or areas.
- Hashed a process model that fit our needs, got Jodied. Stopped process model
altogether.
  Waited for feedback, listened to concerns and addressed them.
- Being spiral: do a little effort, check for feedback, and continue / backup &
reassess
```

```
END File: /Team Documentation/Meeting Notes/note20150411-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150413-Team-Meeting.txt
```

```
2015-04-13 14:10

DECISIONS
Team building @ 12:00, April 25th on campus.
Brand new Java class is XML Reader.
Variables are no longer public; they are private.
Presentation deadline the 27th

ACCOMPLISHMENTS
Thomas s XML Parser is refactored to use public APIs, but not integrated
Thomas refactored the XML Parser
Presentation topic created: How Decisions Affect the Team? This was originally
  How Do Decisions Affect the Team,    but without a metric, it was put down.
Build works on Mac.

UNANSWERED QUESTIONS
What is our message?

TO DO
```

```
- Nate s progress for today.
- Continue exploring accessing exercises in JAR.

RAW NOTES
Refactoring = clean code that everyone can understand.

Exercises need to be placed in .jar.

The team will split
D & T XML
W & M Presentation
J Build
N Exercise Generation

No form is filled out: IDE needs to be used with Notepad to create an XML file.

14:36

15:30

Regarding licenses
JP: When you do work for a class, who owns the work? When you do work for a
university, who owns the work?
Who owns copyright for a class assignment that was created with university
resources?
What is a professor allowed to do with your IP (final exam answers, etc.)
Who owns the IP when something is created in the class?

Only issue is the non-commercial example: if another group of students extends
our
current extension and those students have a grant, and MSUDenver likes it and
wants to
charge a fee to do that, but they can t b/c of licensing fees - is that right?

Do all team members have to agree when granting copying permissions?
Or is it individual control?

Most BlueJ extensions can be decompiled with the Windows Debugger.

Presentations on copyright? Message would be  Should you copyright your
school property?
J: Take care to deliver this pres
T: Message can come out of research
J: Let s try to stick to one idea so we don t move in a circle of ideas
that returns to the first idea.
```

```
END File: /Team Documentation/Meeting Notes/note20150413-Team-Meeting.txt


BEGIN File: /Team Documentation/Meeting Notes/note20150415-Team-Meeting.txt
```

```
015-04-15 14:04

DECISIONS
- Refactoring to be done by end of this week
```

ACCOMPLISHMENTS
- Program takes .xls information to make file list
- Key message vaguely decided
- Extension project compiles again.
- Checklist of user stories uploaded to drive

UNANSWERED QUESTIONS

TO DO
- Hard copy of portfolio to be laid out
- Josh will find an easier way to assemble the portfolio, a plan a.
- Create key message before talking to Jody
- Make checklist of where BlueJ TA is compared to our user stories, on white board.

RAW NOTES

J: Concerned about not enough manhours to complete everything.
Might need java files in project, or just need to say there will be code that makes mapping.
W: Scope was to load an exercise and do an exercise
J:
W: Physical copy to be created, stacked into a PDF. Hard copies to be
N: Are we writing it out on pen?
W: No, we re going to print it out. Easier to print everything and assemble everything out afterwards.
J: Could be a plan b, I might be able to find a better idea

W/D Flesh presentation
T/M/N/J Refactoring / Parser Integration

Check in:
Decided to create a presentation about stakeholders.

J: So this is about outsourcing decision making
D: No, we made the decision

J: No tangible progress - I made a revision on install file for windows (on up yet).
I ve been looking at howing to refactor it
N: You re learning the gaps.

M: Project compiles now on Nate s but not mine.
W: I feel like a lot of our issues are based off building. Make a release folder and put jars in them.

J: Thomas can you start checklist and then we can finish it?
T: I can start it.

T: For user stories that haven t been satisfied, does the framework for it exist becomes the next question.

END File: /Team Documentation/Meeting Notes/note20150415-Team-Meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150418-Team-Meeting.txt

---

2015-04-18 14:01

AGENDA
x Check refactoring progress
x Discuss presentation
? Plan out next few weeks

DECISIONS
- Complete presentation by Wednesday, present on 27th
- Begin work on final presentation on Wednesday 20th
- doc folder inside src folder of project will be deleted.
- Debug meeting after this meeting.
- Inquire about presenting on Wed. 22nd. If that doesn t work, we present on Mon. 27th. Prepare for 22nd regardless of which date is accepted.
- Meet @1:00PM Monday to work / rehearse.

ACCOMPLISHMENTS
- Bought hard materials for project portfolio
- Code stayed same, but moved into folders during refactor.
- Miguel began working on javadoc

UNANSWERED QUESTIONS

TO DO
- Schedule presentation work time between now and Monday, rehearse Monday.

RAW NOTES
Presentation is in two parts: process & project.

M: When is the JUnit test being executed?
N: Check Exercise class for execute method.
M: Exercise not launched. Is it even being called?

J: Javadoc shouldn t be stored in repo.
M: I m working on javadoc in the source code.

J: Make sure errors must be noted. If it s graded right now, I say it deserves a C. What s most important is the individual portfolios. So by Monday we should start shifting gears.
We should get our presentation cranked out too.

W: I originally planned for it to be done on Monday so we could rehearse M/W.

N: Proj. Notebook & Final Presentation can be worked on simultaneously.
W: Would assist each other. Proj. notebook helps final presentation.

J: Monday: .5 hours rehearse
1 - 1.5 hours project prep / inv. portfolio prep.

J: I m using LaTeX.
J: Permission to heavily modify build in Nate s sandbox? I want build file to create jar & move it in extension folder. BlueJ would be responsible for compiling src and stuff.
This is b/c it takes extra time for build to compile.
N: Whole point of build is as an alternative to BlueJ.
J: BlueJ should be doing more of the work.

W: Story or standard intro?

```
T: Story will draw audience in.
W: Vision statement is broad; mission statement is the how. 1st half of our
statement IS vision statement, mission statement includes all of it. We could
tell audience the difference between the two.

W: Does our audience know enough that we can skip the vision statement?
N: I think they created a vision statement, but I  m  not sure.
T: Err on the safe side.

W: Do we want key message to be about vision statement or mission statements?
We created a mission statement, but called it a vision statement
J: Keep the focus on our personal experience, and deliver pertinent information
to keep people on track, and focus on how it will help the audience.

W: If we put information in the handout, we  w o n  t  have enough info to talk
about.

W: Refactor team meeting notes?

n: Proj. Notebook SOFT COPY - Sunday, May 3rd, 4PM.
   Product - Sunday, May 3rd, 4PM.
   Final Presentation - Monday, May 4th.
   Proj. Notebook HARD COPY - Monday, May 4th.
   Ind. Portfolio - Tuesday, May 5th, 3PM.

W:
Mon. 20: @1PM, 1 hr rehearse; 2 hours proj. notebook prep / inv. portfolio prep.
Wed. 22: .5 hours watch Dynamix presenation / 1.5 hrs documentation
Sat. 25: Ind. Portfolios. / Project Notebook / Team Building
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Mon. 27: .5 hours presenting; 1.5 hours
Wed. 29:
Sat. 2:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Sun. 3: Proj. Note. SOFT & Product DUE, 4PM
Mon. 4: Final Pres., Proj. Note. HARD COPY DUE
Tue. 5: Ind. Portfolio DUE
```

END File: /Team Documentation/Meeting Notes/note20150418-Team-Meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150420-Team-Meeting.txt

---

```
2015-04-20 14:04

DECISIONS

ACCOMPLISHMENTS
- Project notebook structure proposed.
- Version 2 of presentation created.
- ToC created in Google Drive
- Monkey tested extensions and logged issues in Redmine.
- Fixed up build.xml so it builds naturally

TO DO
Rehearse presentation.
```

UNANSWERED QUESTIONS

RAW NOTES
T: Do we need to print ALL of our team meeting notes? Print 2 - 3 from beginning, middle & end?
Biggest thing to be done is to run through the source code and check commenting, and to do some testing. We ve done spawn tests, but not full in testing.

Testing: D/W
Presentation: T/N
Artifact Collection: M/J

END File: /Team Documentation/Meeting Notes/note20150420-Team-Meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150422-Team-Meeting.txt

---

2015-04-22 04:28

DECISIONS
- Meet at the outside steps of the Events Center, across from the plaza and Tivoli

ACCOMPLISHMENTS
- Viewed Dynamix s presentation
- Assigned slides to members
- Rehearsed presentation in a circle
- Reviewed logo

TO DO
Next week to team notebook and final presentation.q

UNANSWERED QUESTIONS

RAW NOTES
- Made observations on presentation

JP
Your objective for you presentation is to showcase yourself.
Make your audience emphasize and connect with you.
Keep your focus on the big picture, and the little details with work themselves out.
You ve done good work, now communicate that.
Don t forget which messages are important and which ones are important.

END File: /Team Documentation/Meeting Notes/note20150422-Team-Meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150425-Team-Meeting.txt

---

2015-04-25 12:24

AGENDA
- Rehearse
- Final Presentation
- Project Notebook
- Individual Notebook

DECISIONS
- Final presentation draft to be started tonight
- Everyone will stand fanning out, with three on the left and right.

ACCOMPLISHMENTS
- In BlueJ_Project -> Team Documentation -> Team Portfolio, the ToC and annotations
- Documentation for source code complete
- Began listing topics for final pres
- Rehearsed presentation

TO DO
- Readmes need to be updated.

UNANSWERED QUESTIONS

RAW

W: Pool

M: I started annotations, in drive

T: We re at the point with the outline where we can just fill things out

J So when using LaTeX, there s a build file to build everything. I have a folder mapped to annotations and a folder mapped to the artifacts, and the document can be created by connecting these together.

T I ll probably be writing things up in word. We need someone to port from word to latex

J Latex should be able to import.

M We should be writing in google docs so theres no annotation redundancy.

T Do it all in Miguel s document, because Google Docs can track it. We can pull it down when we re ready to publish it

N We could make assignments

J And it would take time for people to pull it down. If we had a folder layout, it d be easier: everything would be mapped. Biggest concern: how do we get this done, really fast.
Rough draft of final pres should be due this weekend. Final pres rough should be done by the 26th.

T I d rather work on the final when we re done with our presentation.

J But if we finish the rough this weekend, that s less time. I m a little stressed about that. I can start an outline of the final pres, and then we can

revise it.

PROCESS MODEL
- We created user stories, and then code n fixed it.

J I uploaded week by week notebook that we could use as an artifact

T I m going to take our smartboard pictures and make a flow diagram

M What are we working on?

W D o n t stress out out of the documents. The online copy is a placeholder for a hard copy, which is the definitive copy. He wants something online to know that i t s due.

N Should we set up some systems to take care of the presentation. Like a process?

J It needs to be organized.

M List things we want to talk about for the presentation. Must relate to project & process.

J Maybe our process is dichotomy.

W Shoot for 15 minutes. Not 20.

T Does that include demo?

M We should just do it, and however long it is, t h a t s how long it is

D

J L e t s flesh out some topics for the pres.

T 2 of the 5 groups this semester had set roles; in ours we had people thrown wherever.

T We did not adhere to ONE process model. We took bits and pieces of different process models.

END File: /Team Documentation/Meeting Notes/note20150425-Team-Meeting.txt

BEGIN File: /Team Documentation/Meeting Notes/note20150427-Team-Meeting.txt

---

DECISIONS
- The product submission will be a text file pointing to a URL to the repository
- Readmes need to be cleaned up
- Project directories will be moved to TTB format.

ACCOMPLISHMENTS
- Began work on presentation
- Began work on annotations for project portfolio.

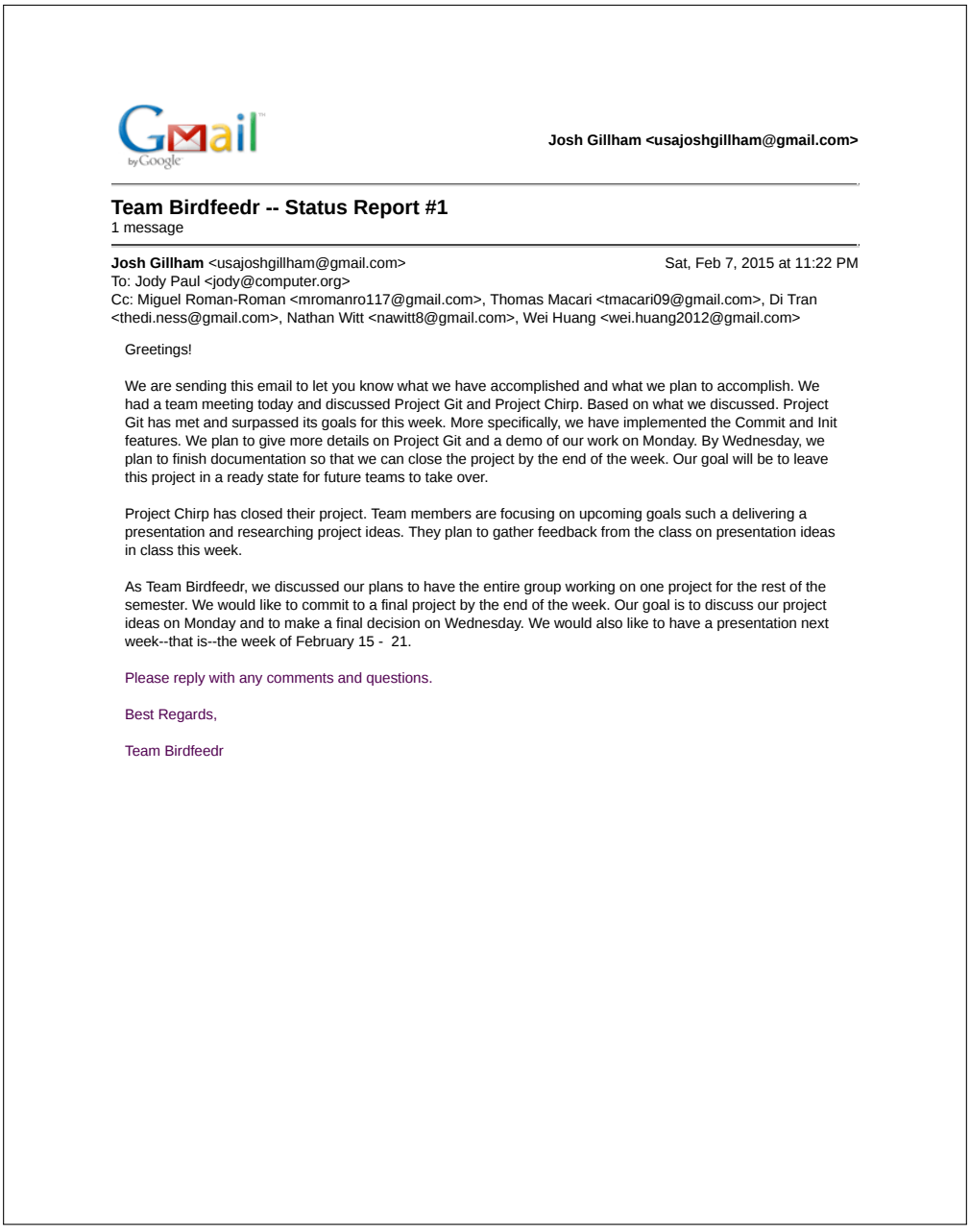
TO DO

```
- Focus on the presentation
- Focus on the project notebook

RAW NOTES
```

END File: /Team Documentation/Meeting Notes/note20150427-Team-Meeting.txt

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

Josh Gillham <usajoshgillham@gmail.com>

Team Birdfeedr -- Status Report #4
2 messages

Wei Huang <wei.huang2012@gmail.com> Mon, Mar 2, 2015 at 4:18 AM
To: Jody Paul <jody@computer.org>, Josh Gillham <usajoshgillham@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, Di Tran <thedi.ness@gmail.com>, N Witt <nawitt8@gmail.com>

Greetings Dr. Paul!

We had a meeting today and made progress. First, we talked about the results of surveys that we conducted with students and teachers on Wednesday. Second, we created a plan of action for this week. Third, we added more ideas to our list of ideas based on the information in the survey. That's what we accomplished today.

We are including our list of possible projects in this email. For the possible projects that you are okay with, we would like to create prototypes for each. On Wednesday, we would like to take our prototypes to show the students. We might need to do that during class time. Would you be okay with us attending the first 30 minutes and then leaving? Also, please let us know if you have any concerns or questions on our list of possible projects.

Best Regards,
Team Birdfeedr

PS List of possible projects. Ideas previously discussed with you in class Wednesday are in purple:

Offline Java API - an extension that has the Java API available while offline. Might include a function that takes a simple sentence and gets the necessary API's for you. (Ex. "I need to create a list" would show the Arraylist API). Addresses student's concerns for syntax.

Thread visualization - an extension that shows what each thread is currently doing as well as variables and interactions with other parts of the code. Addresses student's concerns for visualization of concepts.

Heap visualization - an extension to show the heap and what is currently inside it. Addresses student's concerns for visualization of concepts.

Student networking extension - an extension that would allow students to network with each other while coding. They would be able to message each other and work on the same code. Addresses student's concerns for collaboration.

Better Compiler Error messaging - an extension that would print out simpler error messages. (EX. "File: .../.../code.java: error expected ;" becomes "error on line x. needs a ;" Addresses student's concerns for syntax.

Dr. Jody Paul <jody@computer.org> Tue, Mar 3, 2015 at 11:03 AM
To: Wei Huang <wei.huang2012@gmail.com>
Cc: Josh Gillham <usajoshgillham@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, Di Tran <thedi.ness@gmail.com>, N Witt <nawitt8@gmail.com>

Josh Gillham <usajoshgillham@gmail.com>

Team Birdfeedr -- Status Report #5
8 messages

Josh Gillham <usajoshgillham@gmail.com> Mon, Mar 9, 2015 at 1:43 PM
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 are sending this message to let you know what we have been doing and our accomplishments. The following is in chronological order starting with last Wednesday.

On Wednesday, we oriented Di and Nate back into our team. We also created artifacts towards our project metaphor. Specifically, we created a project definition document, a list of classes, and user stories. For Wednesday, we accomplished more by working concurrently. Thank you for being the catalyst who got us started.

Over the weekend, we created a rough draft of the architecture. This artifact will facilitate communication and will be a part of our project metaphor. We also created a BlueJ extension that only pops up a GUI. On top of those accomplishments, we created a list project ideas.

On our agenda for today, our team will be discussing the rough draft of the architecture to ensure a number of things including that the architecture is not missing any ideas vital to communication along the lines serving as a metaphor. Next, we will try to pick a presentation idea. After that, we will be doing the release and iteration planning. Of course, if we run out of time for these agenda items then we may reschedule.

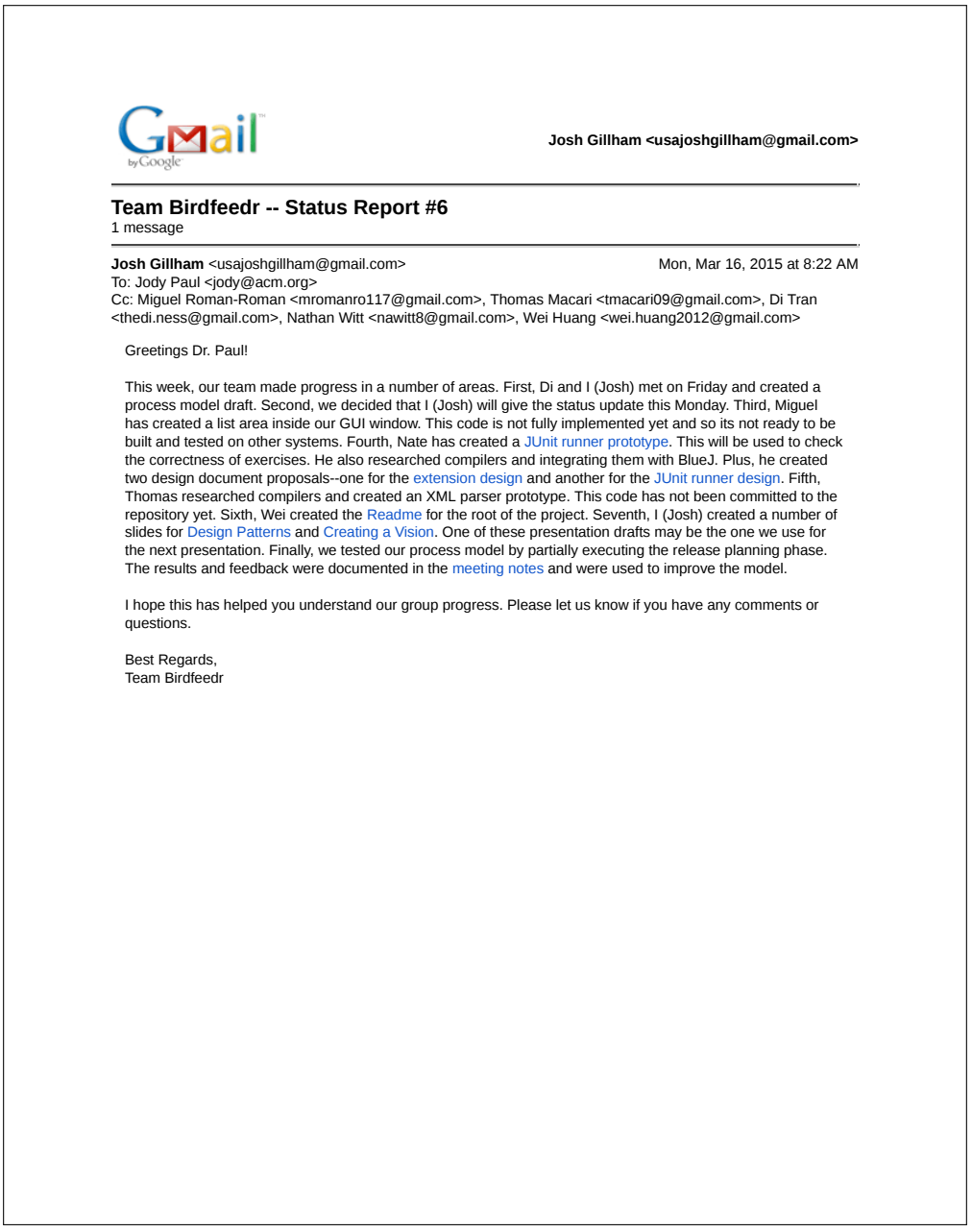
I hope that this status update was illuminating. If there is anything you would like to talk about before class then please feel free to shoot us a reply. See you soon.

Best Regards,
Team Birdfeedr

Dr. Jody Paul <jody@acm.org> Mon, Mar 9, 2015 at 5:39 PM
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>

Thanks for the info. Where would I find the project definition document and user stories? I looked in the repository but didn't see them in any obvious place. --JP
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Mon, Mar 9, 2015 at 11:33 PM
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>



Josh Gillham <usajoshgillham@gmail.com>

Team Birdfeedr -- Status Report #6
1 message

Josh Gillham <usajoshgillham@gmail.com> Mon, Mar 16, 2015 at 8:22 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!

This week, our team made progress in a number of areas. First, Di and I (Josh) met on Friday and created a process model draft. Second, we decided that I (Josh) will give the status update this Monday. Third, Miguel has created a list area inside our GUI window. This code is not fully implemented yet and so its not ready to be built and tested on other systems. Fourth, Nate has created a JUnit runner prototype. This will be used to check the correctness of exercises. He also researched compilers and integrating them with BlueJ. Plus, he created two design document proposals--one for the extension design and another for the JUnit runner design. Fifth, Thomas researched compilers and created an XML parser prototype. This code has not been committed to the repository yet. Sixth, Wei created the Readme for the root of the project. Seventh, I (Josh) created a number of slides for Design Patterns and Creating a Vision. One of these presentation drafts may be the one we use for the next presentation. Finally, we tested our process model by partially executing the release planning phase. The results and feedback were documented in the meeting notes and were used to improve the model.

I hope this has helped you understand our group progress. Please let us know if you have any comments or questions.

Best Regards,
Team Birdfeedr

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Team Birdfeedr Status Report #7

1 message

Di Tran <thedi.ness@gmail.com> Mon, Mar 30, 2015 at 1:09 PM
To: Josh Gillham <usajoshgillham@gmail.com>
Cc: Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>, N Witt <nawitt8@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, Jody Paul <jody@acm.org>

Greetings, Dr. Paul! Here is Team Birdfeedr's weekly status update.

Our group decided to continue working over spring break, and we have had many accomplishments. First, we created a class that allows the GUI to know which exercise the user is working in. Second, we redesigned the GUIs so that the right click menu options only show up on exercises - it's not fully implemented, but it is designed. Next, we simplified our JUnit Test Runner component for easier integration into the project. The first iteration of the project is complete and ready to be seen! To wrap things up, we added a new policy to screen share team meeting notes while they are being taken.

We concurrently developed our presentation, and we are nearly done.

Team Birdfeedr

On Mar 30, 2015 11:48 AM, "Josh Gillham" <usajoshgillham@gmail.com> wrote:

> This is much more understandable. Ty
>
> On Mar 30, 2015 10:38 AM, "Di Tran" <thedi.ness@gmail.com> wrote:
>
> > Thanks for the feedback! Here is the new draft:
> >
> > ~~~~~~~~~~~~~~~~
> >
> > Greetings, Dr. Paul! Here is Team Birdfeedr's weekly status update.
> >
> > Our group decided to continue working over spring break, and we have had many accomplishments. First, we created a class that allows the GUI to know which exercise the user is working in. Second, we redesigned the GUIs so that the right click menu options only show up on exercises - it's not fully implemented, but it is designed. Next, we simplified our JUnit Test Runner component for easier integration into the project. The first iteration of the project is complete and ready to be seen! To wrap things up, we added a new policy to screen share team meeting notes while they are being taken.
> >
> > We concurrently developed our presentation, and we are nearly done.
> >
> > Team Birdfeedr
> >
> > On Mon, Mar 30, 2015 at 2:39 AM, Josh Gillham <usajoshgillham@gmail.com> wrote:
> >
> > > Thanks for your input, Wei.
> > >
> > > Everybody, let's keep up our focus. We are almost done with this semester. I would like to finish strong. On the other hand, I would feel ashamed if we start making trivial mistakes now. So let's do our

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Team Birdfeedr Status Report #8

2 messages

Di Tran <thedi.ness@gmail.com> Mon, Apr 6, 2015 at 11:20 AM
To: Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>, Jody Paul <jody@acm.org>

Greetings, Dr. Paul!

This message will talk about Birdfeedr's progress over the past week.

Our week was highlighted by our presentation, with several people working on it at different times. We rehearsed, presented and reflected on it on Wednesday, and are planning to have another presentation date for the 15th and the 29th. However, we have made other progress as well.

Our parser was integrated with the rest of our extensions. We discovered caching issues where an exercise became cached in the jar. We've fixed that issue. We also made the decision to design our backend in more detail, and are currently looking to answer these two questions: "Where do exercise files live?" and "What happens after a user clicks the load button?" We've set an internal deadline for April 11th to solve the second question.

In addition, we have begun cross training team members to do different parts of this project's workload. For example, Josh's documentation role has shifted to Di. We intend on doing more of this during this week.

Team BirdFeedr

Dr. Jody Paul <jody@acm.org> Mon, Apr 6, 2015 at 1:13 PM
Reply-To: jody@acm.org
To: Di Tran <thedi.ness@gmail.com>
Cc: Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Sounds great. Thank you.

You might consider the location of exercises as being a settable item in the Preferences > Extensions interface. This could accept a local directory or a URI that could be in the cloud.

--JP

[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com>

Team Birdfeedr Status Report #8
1 message

Wei Huang <wei.huang2012@gmail.com> Mon, Apr 13, 2015 at 11:52 AM
To: Jody Paul <jody@computer.org>, 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>

Greetings, Dr. Paul!

This message will talk about Birdfeedr's progress over the past week.

As a group we met our internal deadline of April 11th to solve the 2 questions: "Where do exercise files live?" and "What happens after a user clicks the load button?". We decided that the exercises files will be located at a user specified location using the extension's tab in BlueJ preferences. We also decided that after clicking the load button, a new project is created with the chosen exercise.

We also changed our meeting notes format from a stenographer format to a meeting minute's format.

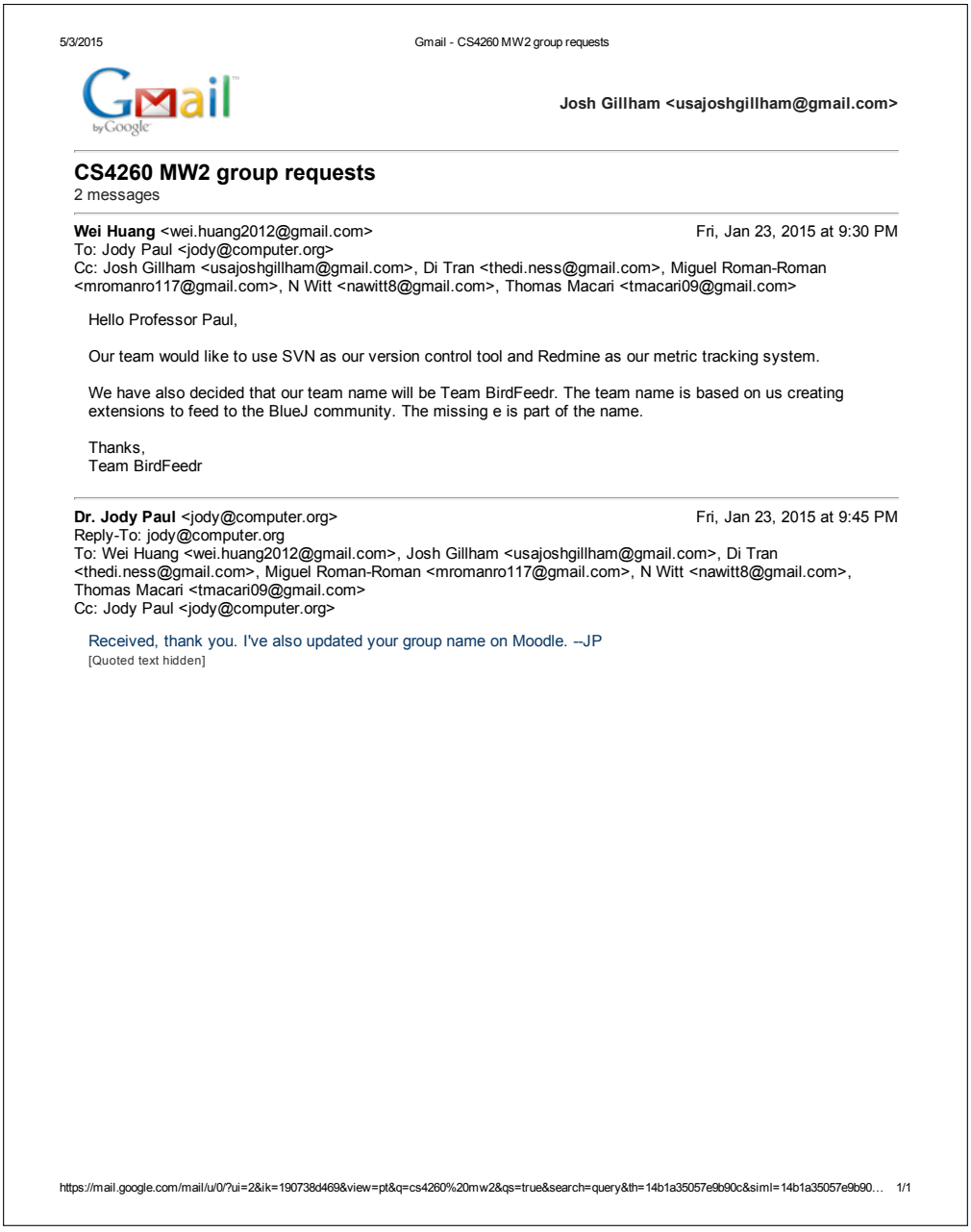
We reflected on our past experiences and what worked for us to find a good message for the presentation.

Finally, we decided to give our project a name, and the name we decided on was "BlueJ TA". We chose this name because a teacher's assistant supports students and instructors which is what we were aiming for with our extension.

Sincerely,
Team BirdFeedr

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]

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

QA Update

4 messages

Di Tran <thedi.ness@gmail.com> Tue, Feb 24, 2015 at 9:12 PM
To: Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

I'm going to follow up Josh with some of my own thoughts regarding the scripts and powerpoint.

The powerpoint looks great. No wall of text, and everything looks neat and to the point. I am unable to check to see if the bullet points animate (Powerpoint crashes on my computer), but the rest of it looks good to go!

For the narrator scripts, I am concerned that it will sound too.. scripted, as opposed to coming out naturally. Not sure it can be remedied between now and tomorrow, but it's something I hope will go smoothly.

Reminder: We'll meet in the classroom at 1:30 tomorrow!

Di

Josh Gillham <usajoshgillham@gmail.com> Tue, Feb 24, 2015 at 9:33 PM
To: Di Tran <thedi.ness@gmail.com>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

I am running Word 2013 and it animates. The 2nd slide with the paragraph might be hard for the audience to read. See everybody tomorrow.
-Josh
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Tue, Feb 24, 2015 at 9:35 PM
To: Di Tran <thedi.ness@gmail.com>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Never mind about the paragraph.
-Josh
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Tue, Feb 24, 2015 at 9:55 PM
To: Di Tran <thedi.ness@gmail.com>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Most everything looks great. Thanks to everyone who contributed tonight. We just need those stronger transitions and we should be ready to go.
-Josh
[Quoted text hidden]

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Extension Name Idea!

2 messages

Di Tran <thedi.ness@gmail.com> Wed, Mar 4, 2015 at 8:32 PM
To: Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Hey all,

Someone mentioned that our extension / main project needed a name. What do you guys think of the name "Code 'N' Check"?

Di

Wei Huang <wei.huang2012@gmail.com> Wed, Mar 4, 2015 at 9:00 PM
To: Di Tran <thedi.ness@gmail.com>
Cc: Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>

Programming exercises extension
Coding practice via BlueJ extensions
CodingBird
BlueBat
BlueB
BirdFeedr
A programmer's guide to the galaxy

Just a few names.

As for Code N Check. I'm alright with it. I have nothing for or against it honestly.
[Quoted text hidden]

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Wiki Update

3 messages

Di Tran <thedi.ness@gmail.com> Tue, Mar 10, 2015 at 10:10 PM
To: Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Hey, guys!

I've updated the Wiki page: we now have a specific page for our extension. You can find it on our main wiki, underneath the heading "Current Status." So far, I've only ported our project definition and tweaked some of the wording. Once we get SVN restructured, I can start linking to specific resources in our repository, and talk more about the project.

Feel free to make suggestions about the wiki!

Di

Josh Gillham <usajoshgillham@gmail.com> Tue, Mar 10, 2015 at 10:21 PM
To: Di Tran <thedi.ness@gmail.com>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Looks like a good start. Don't forget to add JP to the stake holders.
-Josh
[Quoted text hidden]

Di Tran <thedi.ness@gmail.com> Tue, Mar 10, 2015 at 10:25 PM
To: Josh Gillham <usajoshgillham@gmail.com>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Thanks, I just added it!
[Quoted text hidden]

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Tailored Process Model Draft (draft)

8 messages

Josh Gillham <usajoshgillham@gmail.com> Fri, Mar 13, 2015 at 8:45 AM
To: Di Tran <thedi.ness@gmail.com>

Okay, Di, here is the draft message. Let me know what you think.
-Josh
--
Greetings Dr. Paul!

After class on Wednesday, our group thought we needed to spend some time to tailor a process model for the current project. Di and I (Josh) came up with a draft for the process model and we wanted to run it by you to see if you have suggestions for how to make it better. We are expecting to make another revision to this before our team meeting tomorrow. Please let us know what you think.

Best Regards,
Di Tran
Josh Gillham

Josh Gillham <usajoshgillham@gmail.com> Fri, Mar 13, 2015 at 8:49 AM
To: Di Tran <thedi.ness@gmail.com>

I forgot the picture.

[Quoted text hidden]

20150312_155046.jpg
1994K

Di Tran <thedi.ness@gmail.com> Fri, Mar 13, 2015 at 11:06 AM
To: Josh Gillham <usajoshgillham@gmail.com>

Do you think we should consult with each other tonight and the team before we send it to Jody? This process model serves us more than it serves him and the team would be able to give better feedback because we're the ones using it the most.

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Wednesday Agenda?

7 messages

Di Tran <thedi.ness@gmail.com> Mon, Mar 30, 2015 at 9:58 PM
To: Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Hi everybody!

So Miguel and I were talking and we were thinking that Wednesday's agenda be about rethinking the design of the backend of the program. What do you all think of that?

Di

N Witt <nawitt8@gmail.com> Tue, Mar 31, 2015 at 12:06 AM
To: Di Tran <thedi.ness@gmail.com>
Cc: Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

A design might be good, the front end got redesigned with all the new changes so the back end would haft to be as well.
[Quoted text hidden]

Wei Huang <wei.huang2012@gmail.com> Tue, Mar 31, 2015 at 5:31 AM
To: N Witt <nawitt8@gmail.com>
Cc: Di Tran <thedi.ness@gmail.com>, Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>

sounds like a good agenda to me.

so the layout of the day will look something like:

1) Presentations prep.
2) Presentations
3) break?
4) backend design
5) plan next team building activity.
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Tue, Mar 31, 2015 at 1:19 PM
To: Wei Huang <wei.huang2012@gmail.com>
Cc: N Witt <nawitt8@gmail.com>, Di Tran <thedi.ness@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>

I think that's fine, but, what about reporting on the progress we have made?
-Josh

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Remember to upload presentation deliverables

8 messages

Josh Gillham <usajoshgillham@gmail.com> Wed, Apr 1, 2015 at 4:21 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 just wanted to make sure that we upload presentation deliverables as a team. Of course, we should probably agree what those are... Please treat this like a thread and carry the conversation along.

Best Regards,
Josh Gillham

Wei Huang <wei.huang2012@gmail.com> Thu, Apr 2, 2015 at 3:11 AM
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>

I'd say everything in presentation 2 folder minus maybe the bank example model, and Di's notes. I say maybe because it was part of the presentation/ making of it so it's up to Di if he wants to keep it in or not.
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Thu, Apr 2, 2015 at 8:08 AM
To: Wei Huang <wei.huang2012@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>

I assume you are saying convert the GDoc into a PDF.
[Quoted text hidden]

Di Tran <thedi.ness@gmail.com> Thu, Apr 2, 2015 at 8:28 AM
To: Josh Gillham <usajoshgillham@gmail.com>
Cc: Wei Huang <wei.huang2012@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Nathan Witt <nawitt8@gmail.com>

Those notes aren't integral to the presentation. So when the presentation is submitted, zip up:
- Design Pattern slides
- Works Cited doc
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Thu, Apr 2, 2015 at 8:51 AM
To: Di Tran <thedi.ness@gmail.com>
Cc: Wei Huang <wei.huang2012@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Nathan Witt <nawitt8@gmail.com>

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Team Building

5 messages

Josh Gillham <usajoshgillham@gmail.com> Wed, Apr 1, 2015 at 4:22 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!

This new thread is for planning a team building exercise.

Best Regards,
Josh Gillham

Di Tran <thedi.ness@gmail.com> Wed, Apr 1, 2015 at 6:35 PM
To: Josh Gillham <usajoshgillham@gmail.com>
Cc: Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>

I would like all of us to go out and do something again. Either eating out or going bowling or something along those lines.

[Quoted text hidden]

Wei Huang <wei.huang2012@gmail.com> Thu, Apr 2, 2015 at 3:07 AM
To: Di Tran <thedi.ness@gmail.com>
Cc: Josh Gillham <usajoshgillham@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>

how about an arcade?
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Thu, Apr 2, 2015 at 8:08 AM
To: Wei Huang <wei.huang2012@gmail.com>
Cc: Di Tran <thedi.ness@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>

Ya, arcades are cool.
-Josh
[Quoted text hidden]

Di Tran <thedi.ness@gmail.com> Thu, Apr 2, 2015 at 8:27 AM
To: Josh Gillham <usajoshgillham@gmail.com>
Cc: Wei Huang <wei.huang2012@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Nathan Witt <nawitt8@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Team Birdfeedr -- Custom Tailored Process Model Draft

4 messages

Josh Gillham <usajoshgillham@gmail.com> Sun, Mar 15, 2015 at 4:34 PM
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!

From the conversation we had with you last class period, we wanted to organize the processes that we have been using on previous projects into a formal process model to increase our organization and transparency. Furthermore, this formal process model will help us do a better job of keeping you in the loop. We would like to get your feedback on our formal process model draft. You can find that document here.

Your standard format of giving us feedback by quoting the document will be great. Please reply with any comments, questions, and your feedback of course. We don't consider this document complete and we are committed to improving on it in the future. Thank you for your help in advance!

Best Regards,
Team Birdfeedr

Dr. Jody Paul <jody@acm.org> Mon, Mar 16, 2015 at 9:53 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>

It looks like much effort went into this draft.
There's a lot going on and it is likely that I'm not sharing the same interpretation.
Probably best for us to dedicate 15 minutes during class on Tuesday for a walkthrough.
--JP
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Mon, Mar 16, 2015 at 1:26 PM
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>

Do you mean today or Wednesday? I would be happy to spend some time in class on it, but, I am not speaking for everyone on our team.
-Josh
[Quoted text hidden]

Dr. Jody Paul <jody@acm.org> Mon, Mar 16, 2015 at 1:48 PM

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Handout Editing

3 messages

Di Tran <thedi.ness@gmail.com> Sun, Apr 19, 2015 at 3:05 PM
To: Josh Gillham <usajoshgillham@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>, N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Wei Huang <wei.huang2012@gmail.com>

Hey all!

I've begin doing some editing on the handout. It's a bit of a rehaul from what was there before, so could I get some feedback before I continue going in this direction?

Josh Gillham <usajoshgillham@gmail.com> Sun, Apr 19, 2015 at 11:47 PM
To: Di Tran <thedi.ness@gmail.com>

I'll take a look at it tomorrow morning.
-Josh
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Mon, Apr 20, 2015 at 8:35 AM
To: Di Tran <thedi.ness@gmail.com>

I checked the handout and fixed minor issues. Over all it looks complete.
-Josh
[Quoted text hidden]

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Team building tomorrow

7 messages

Miguel Roman-Roman <mromanro117@gmail.com> Fri, Apr 24, 2015 at 8:34 PM
To: N Witt <nawitt8@gmail.com>, Josh Gillham <usajoshgillham@gmail.com>, Wei Huang <wei.huang2012@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>

Hey guys,
What time did we say we were gona meet for the team building?

--Miguel

Di Tran <thedi.ness@gmail.com> Fri, Apr 24, 2015 at 8:39 PM
To: Miguel Roman-Roman <mromanro117@gmail.com>
Cc: Thomas Macari <tmacari09@gmail.com>, Josh Gillham <usajoshgillham@gmail.com>, Wei Huang <wei.huang2012@gmail.com>, N Witt <nawitt8@gmail.com>

12 pm

[Quoted text hidden]

Wei Huang <wei.huang2012@gmail.com> Fri, Apr 24, 2015 at 10:01 PM
To: Di Tran <thedi.ness@gmail.com>
Cc: Miguel Roman-Roman <mromanro117@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Josh Gillham <usajoshgillham@gmail.com>, N Witt <nawitt8@gmail.com>

Yeap we are meeting at the outside steps at the Tivoli front entrance next to the student success building and king's center.
[Quoted text hidden]

Di Tran <thedi.ness@gmail.com> Fri, Apr 24, 2015 at 10:19 PM
To: Wei Huang <wei.huang2012@gmail.com>
Cc: Josh Gillham <usajoshgillham@gmail.com>, Thomas Macari <tmacari09@gmail.com>, N Witt <nawitt8@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>

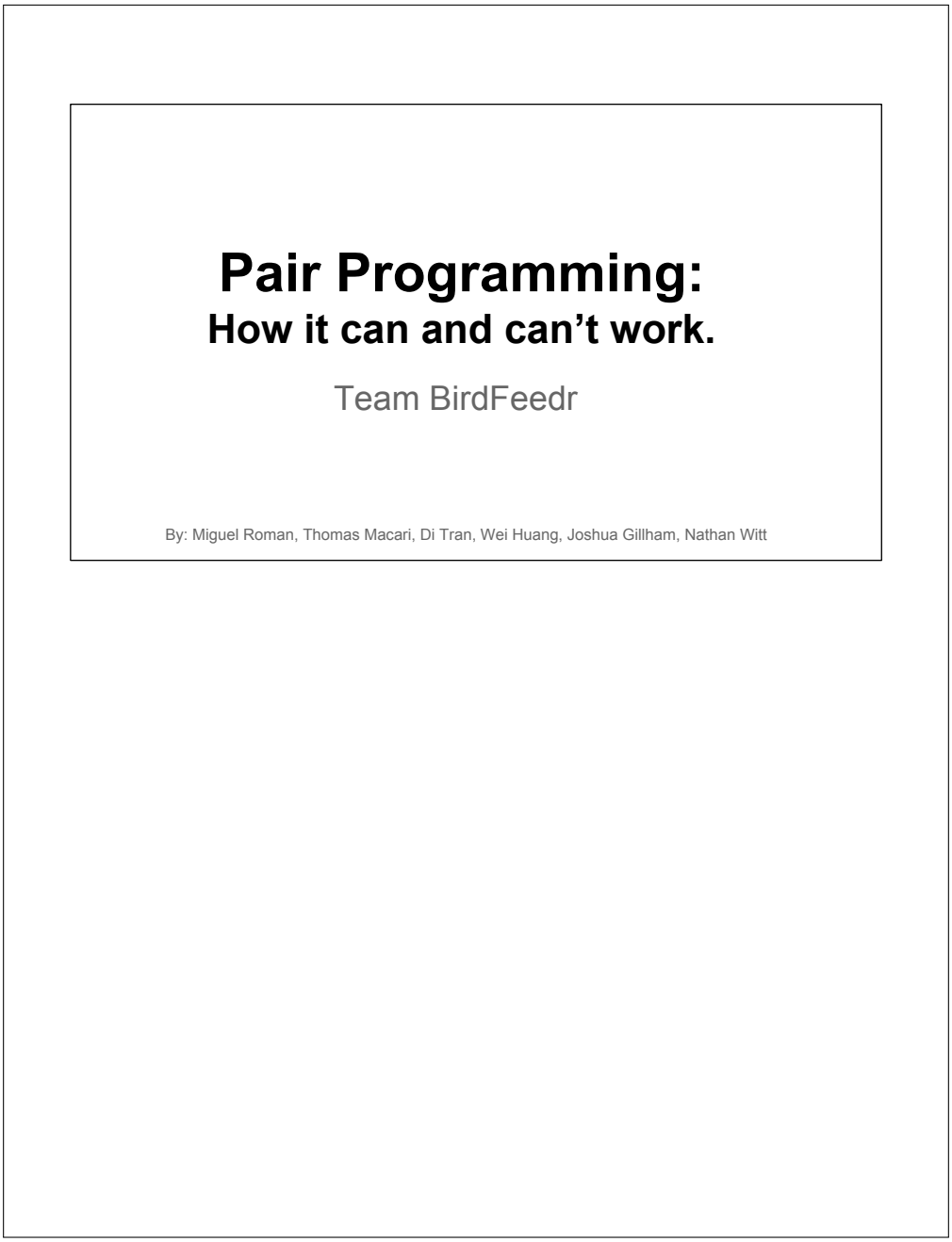
Wait, so not the event place steps?

[Quoted text hidden]

Di Tran <thedi.ness@gmail.com> Fri, Apr 24, 2015 at 10:19 PM
To: Wei Huang <wei.huang2012@gmail.com>
Cc: Thomas Macari <tmacari09@gmail.com>, Josh Gillham <usajoshgillham@gmail.com>, N Witt <nawitt8@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>

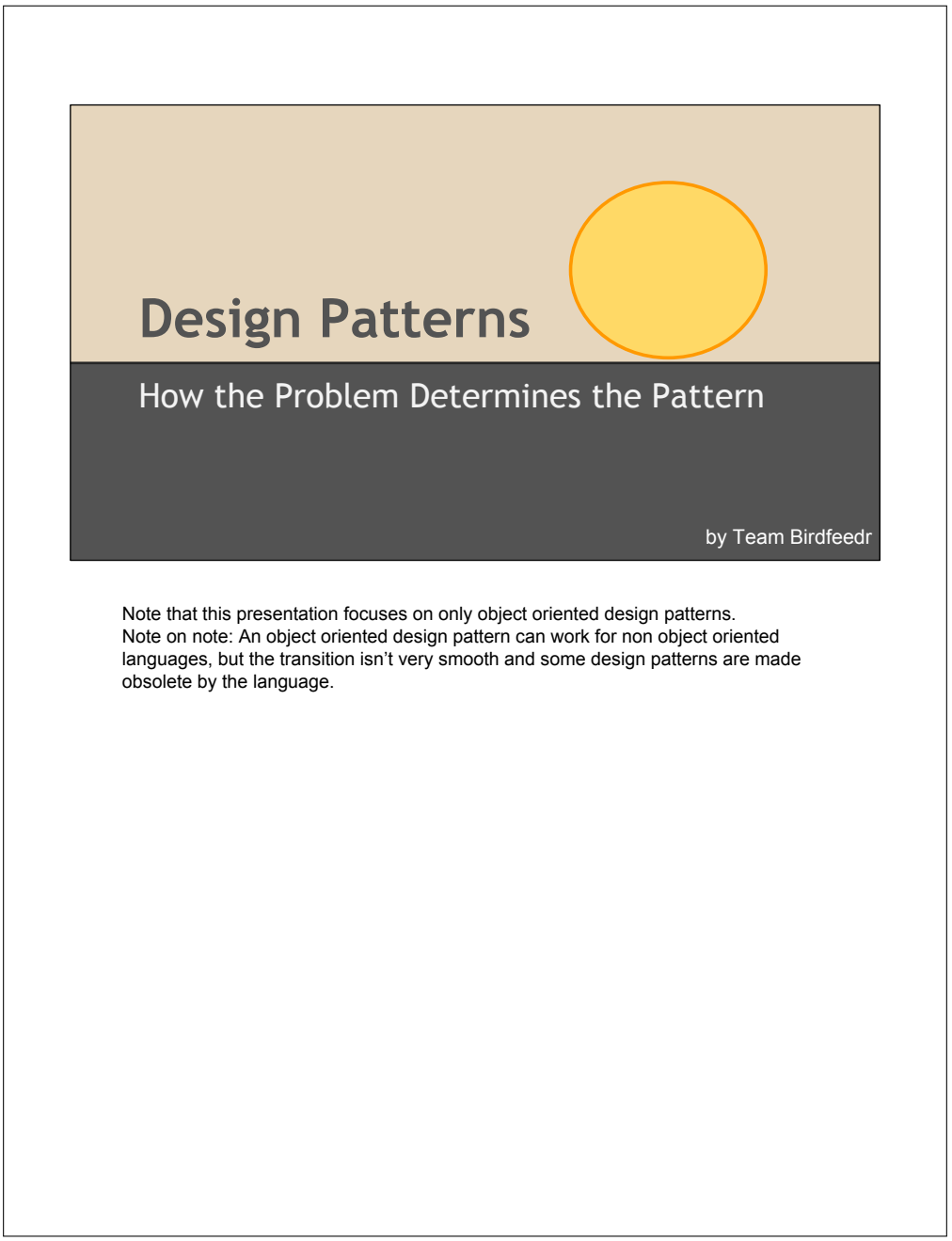
Presentation 1 Explains Pair Programming

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

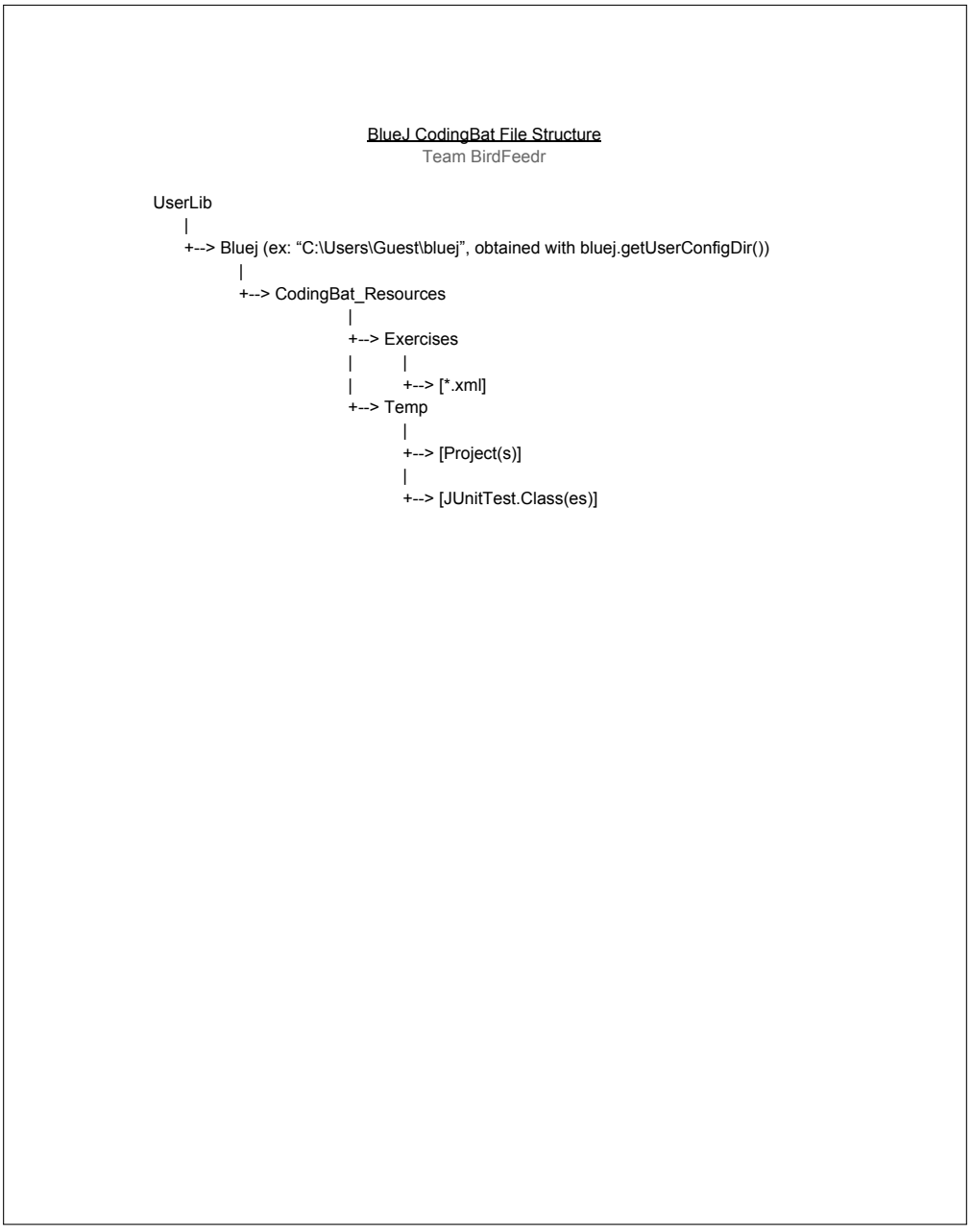
Description The final presentation talks about our work throughout the semester. First, we talk about the process we went through to develope the product. Last, we talk about the product by demoing the extension and explaining what happens behind the scenes.

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
```

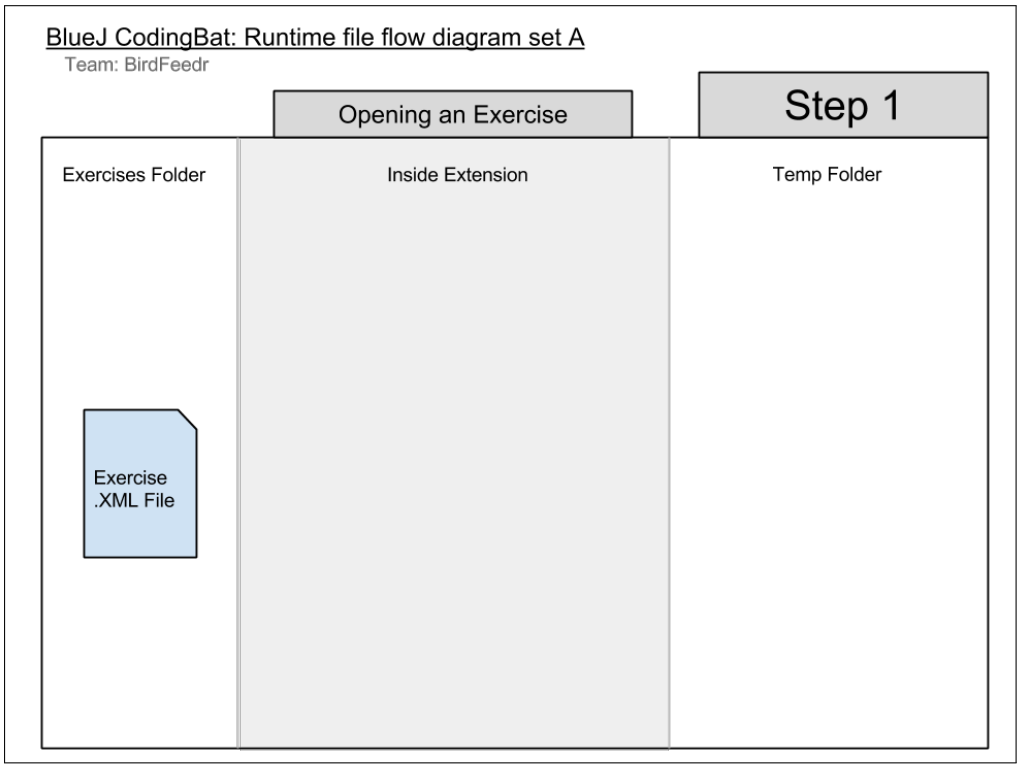
```
public class Adder {

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

}
```

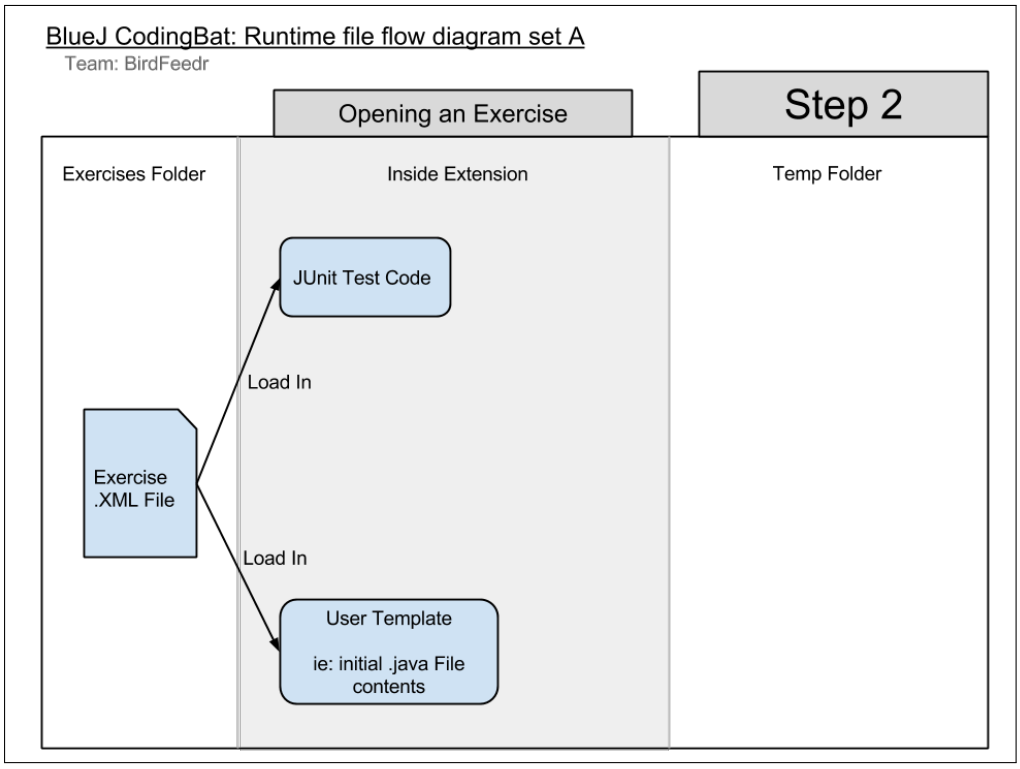
```
END File:
/Sandbox/Nate/BackEnd_FileFlow_Proposals/FileStructure/FileStructure_Example/Blu
eJCodingBat_Resources/Temp/UserProject/Adder.java
```

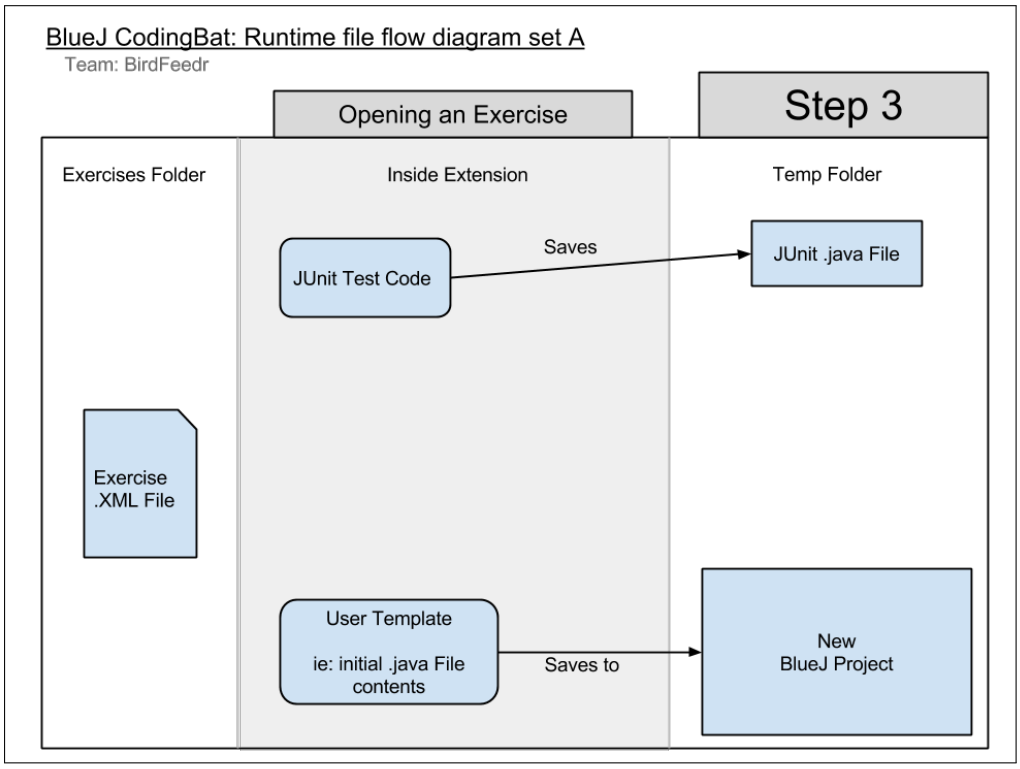
```
BEGIN File:
/Sandbox/Nate/BackEnd_FileFlow_Proposals/FileStructure/FileStructure_Example/Blu
eJCodingBat_Resources/Temp/UserProject/README.TXT
```

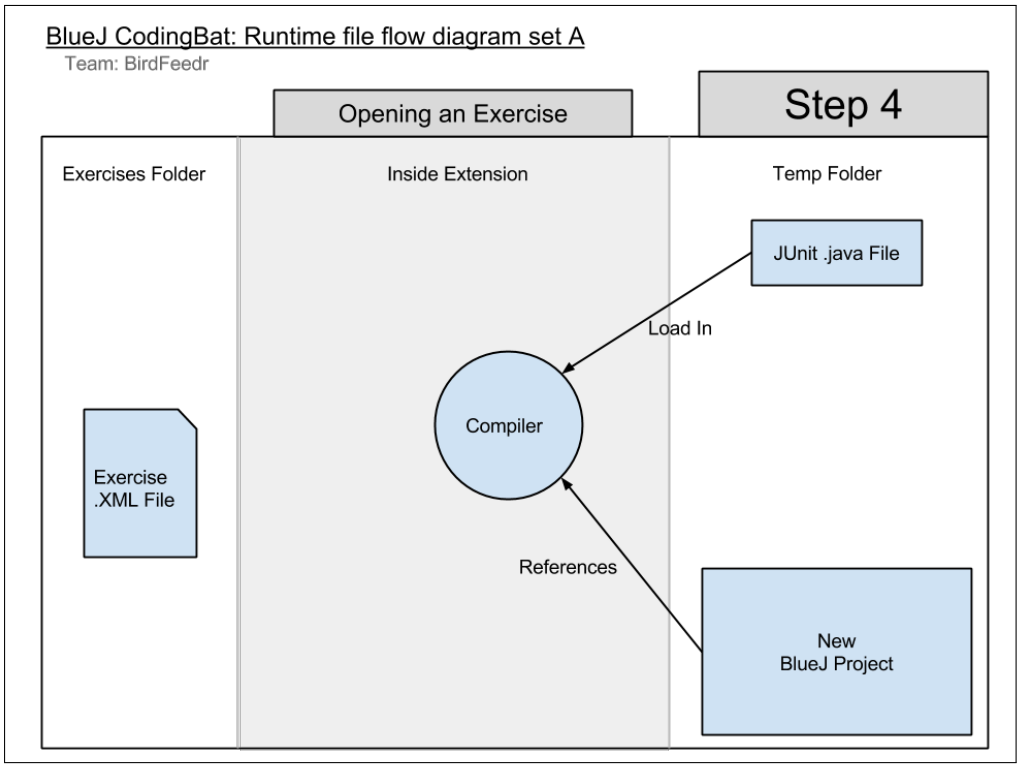
```
------------------------------------------------------------------------
This is the project README file. Here, you should describe your project.
Tell the reader (someone who does not know anything about this project)
all he/she needs to know. The comments should usually include at least:
------------------------------------------------------------------------

PROJECT TITLE:
PURPOSE OF PROJECT:
VERSION or DATE:
HOW TO START THIS PROJECT:
AUTHORS:
USER INSTRUCTIONS:
```

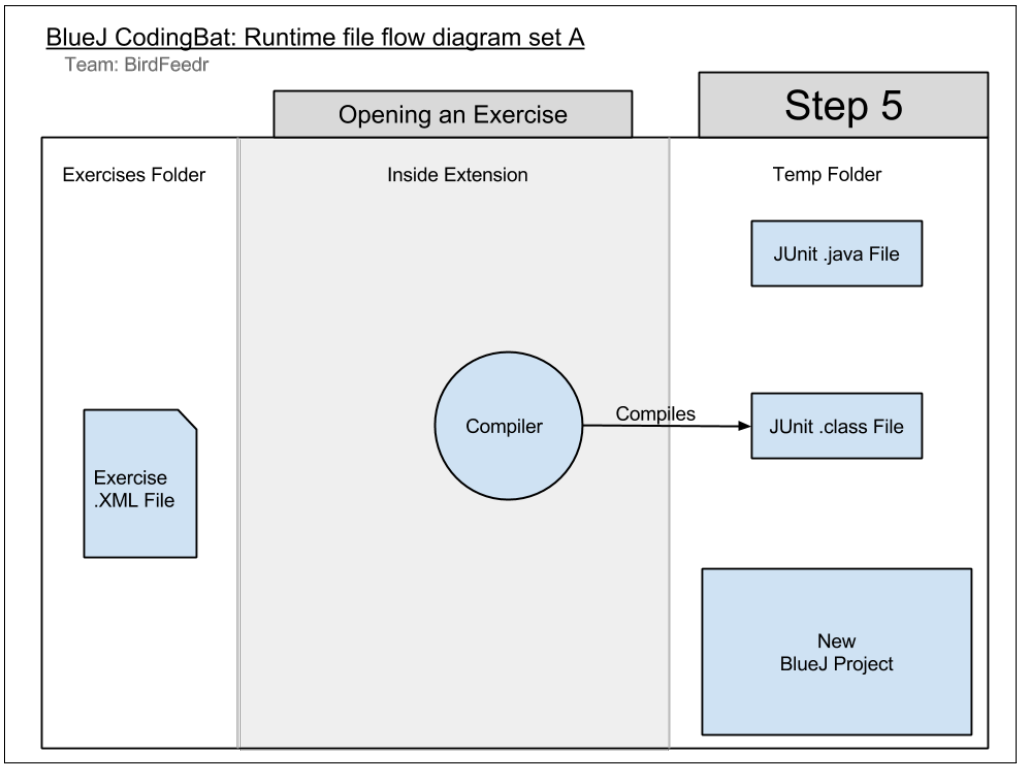
```
END File:
/Sandbox/Nate/BackEnd_FileFlow_Proposals/FileStructure/FileStructure_Example/Blu
eJCodingBat_Resources/Temp/UserProject/README.TXT
```

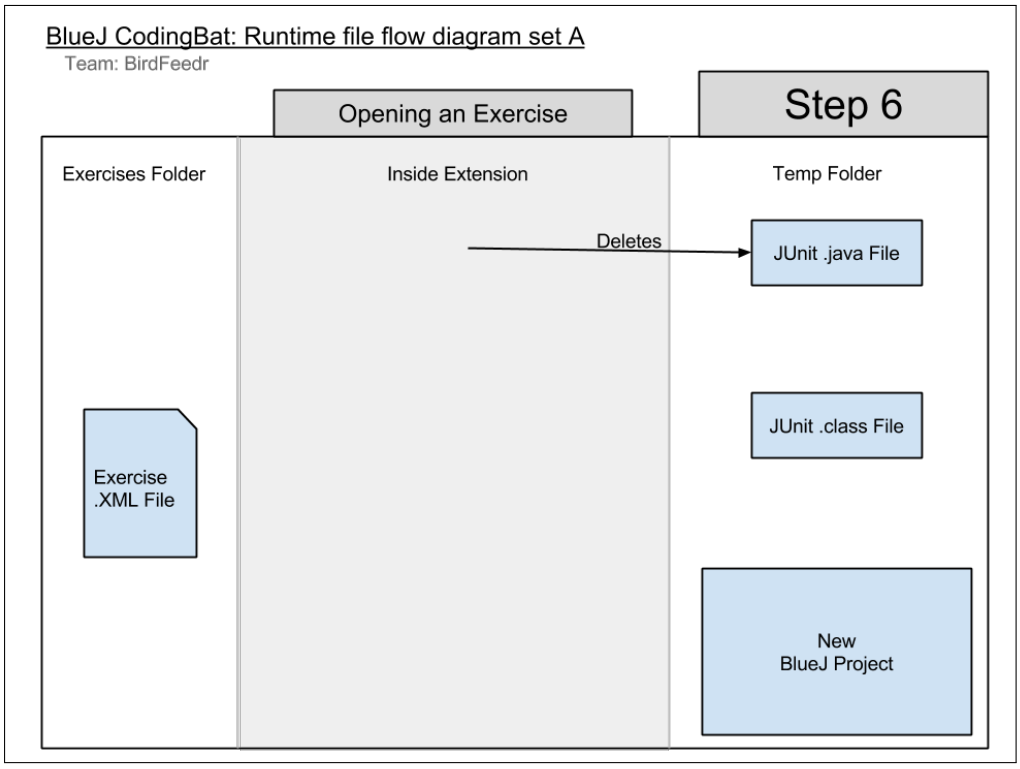

BlueJ CodingBat: Runtime file flow diagram set A
Team: BirdFeedr
Opening an Exercise
Step 1
Exercises Folder
Inside Extension
Temp Folder
Exercise
.XML File

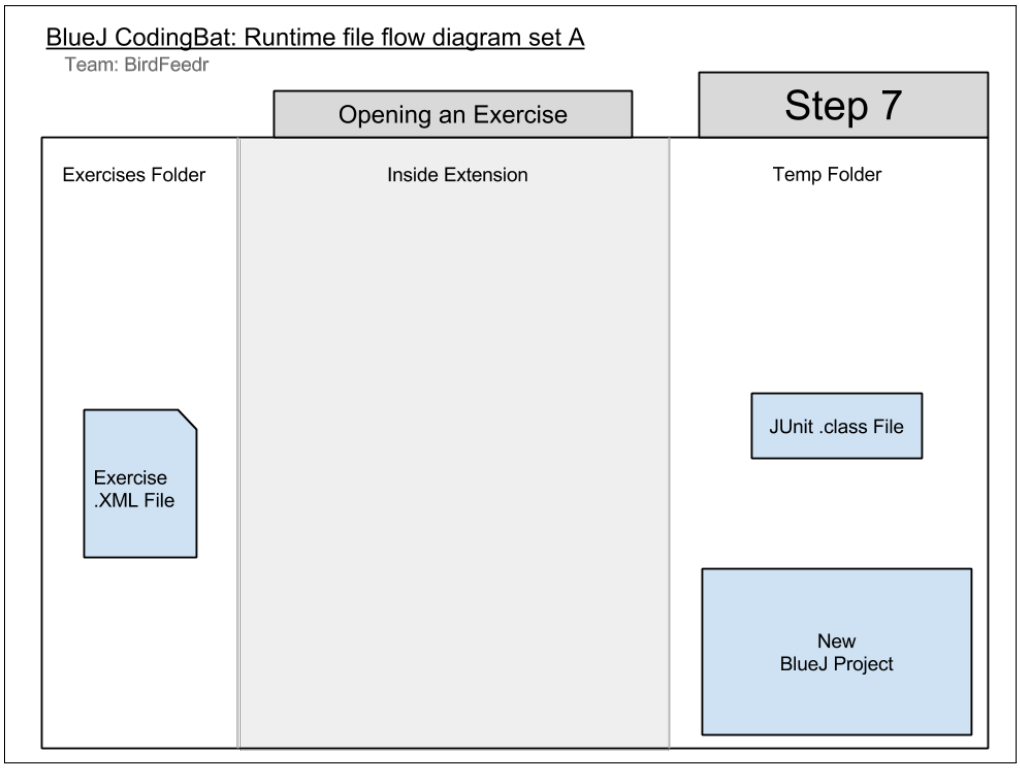

BlueJ CodingBat: Runtime file flow diagram set A
Team: BirdFeedr
Opening an Exercise
Step 2
Exercises Folder
Inside Extension
Temp Folder
JUnit Test Code
Load In
Exercise
.XML File
Load In
User Template
ie: initial .java File
contents


BlueJ CodingBat: Runtime file flow diagram set A
Team: BirdFeedr
Opening an Exercise
Step 3
Exercises Folder
Inside Extension
Temp Folder
JUnit Test Code
Saves
JUnit .java File
Exercise
.XML File
User Template
ie: initial .java File
contents
Saves to
New
BlueJ Project


BlueJ CodingBat: Runtime file flow diagram set A
Team: BirdFeedr
Opening an Exercise
Step 4
Exercises Folder
Inside Extension
Temp Folder
JUnit .java File
Load In
Compiler
Exercise
.XML File
References
New
BlueJ Project


BlueJ CodingBat: Runtime file flow diagram set A
Team: BirdFeedr
Opening an Exercise
Step 5
Exercises Folder
Inside Extension
Temp Folder
JUnit .java File
Compiler
Compiles
JUnit .class File
Exercise
.XML File
New
BlueJ Project


BlueJ CodingBat: Runtime file flow diagram set A
Team: BirdFeedr
Opening an Exercise
Step 6
Exercises Folder
Inside Extension
Temp Folder
Deletes
JUnit .java File
JUnit .class File
Exercise
.XML File
New
BlueJ Project


BlueJ CodingBat: Runtime file flow diagram set A
Team: BirdFeedr
Opening an Exercise
Step 7
Exercises Folder
Inside Extension
Temp Folder
Exercise
.XML File
JUnit .class File
New
BlueJ Project

Proposed BlueJ TA Design for first Iteration

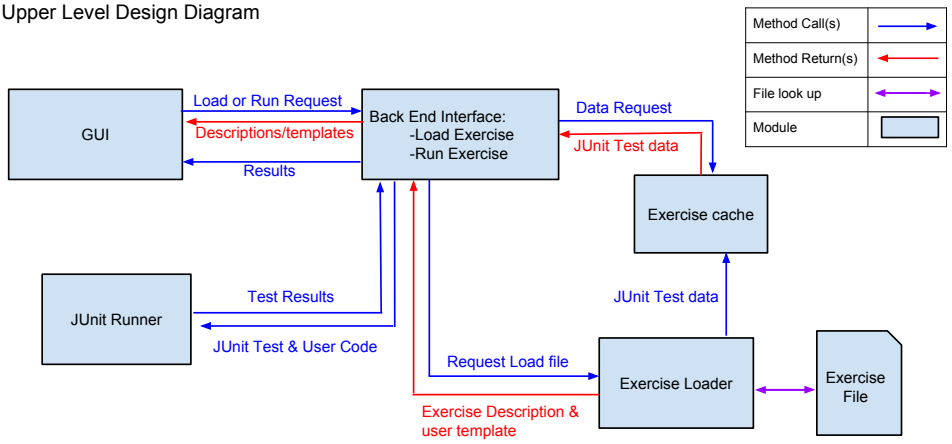
Description Design proposal contains upper level design diagram of how modules will link and communicate with each other on an upper level. We created this diagram to help us better understand how the extension will work and what needed to be done to create the first iteration as well as how we could 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:

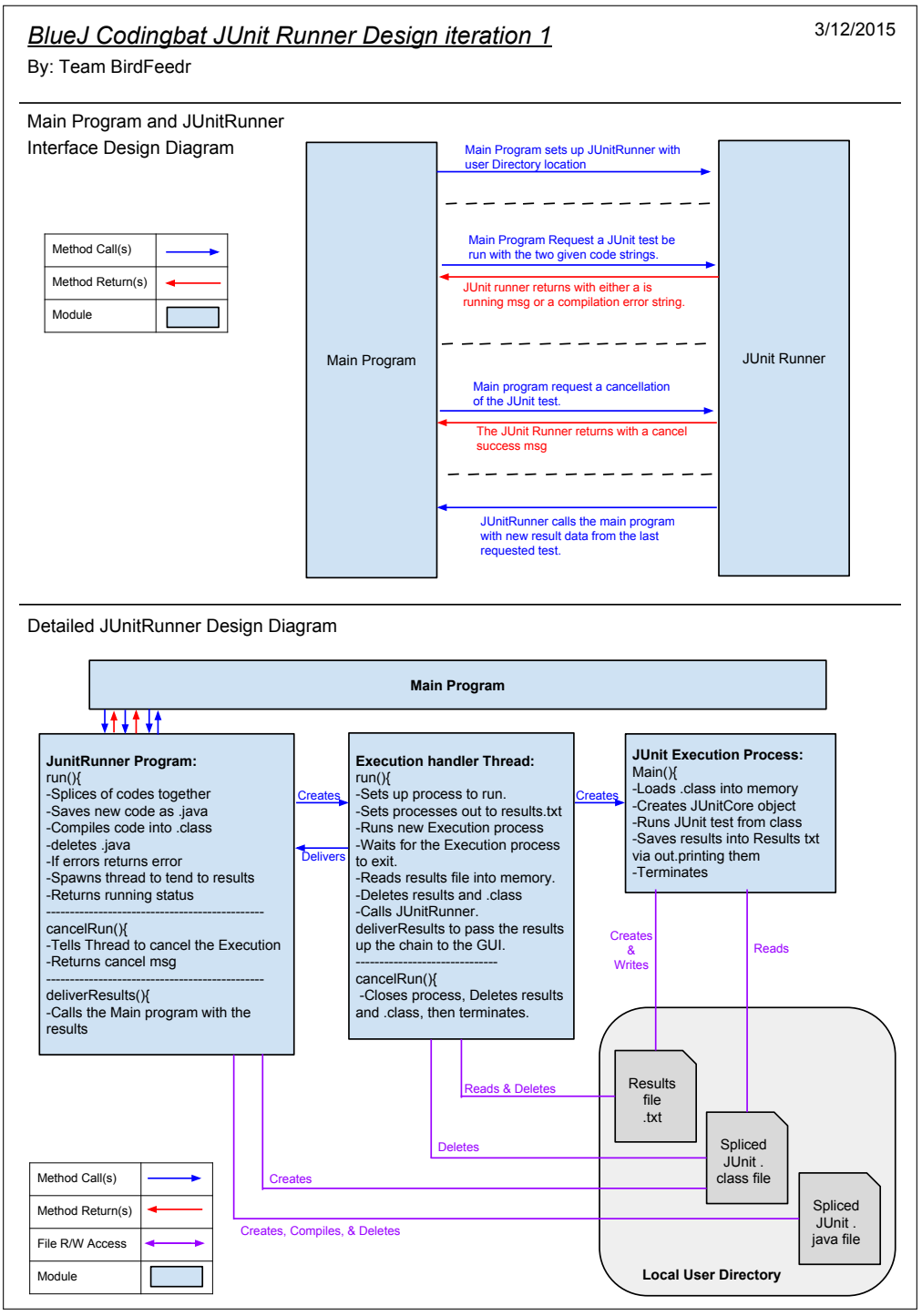
A. GUI request Excersize located at file "F" to be loaded from the interface.
B. Interface passes the request to the Exercise loader, and waits for a response.
C. The exercise loader locates the file and parses its contents into memory.
D. The Exercise parser stores the JUnit data in the Exercise cash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

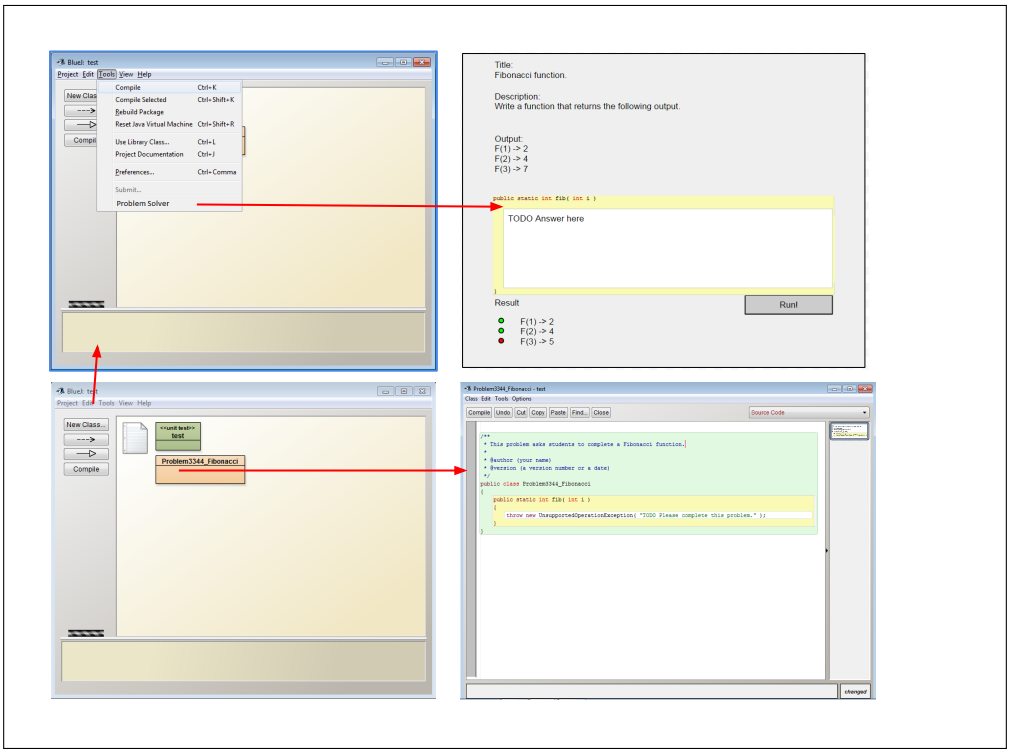
Proposed JUnitRunner Design v1

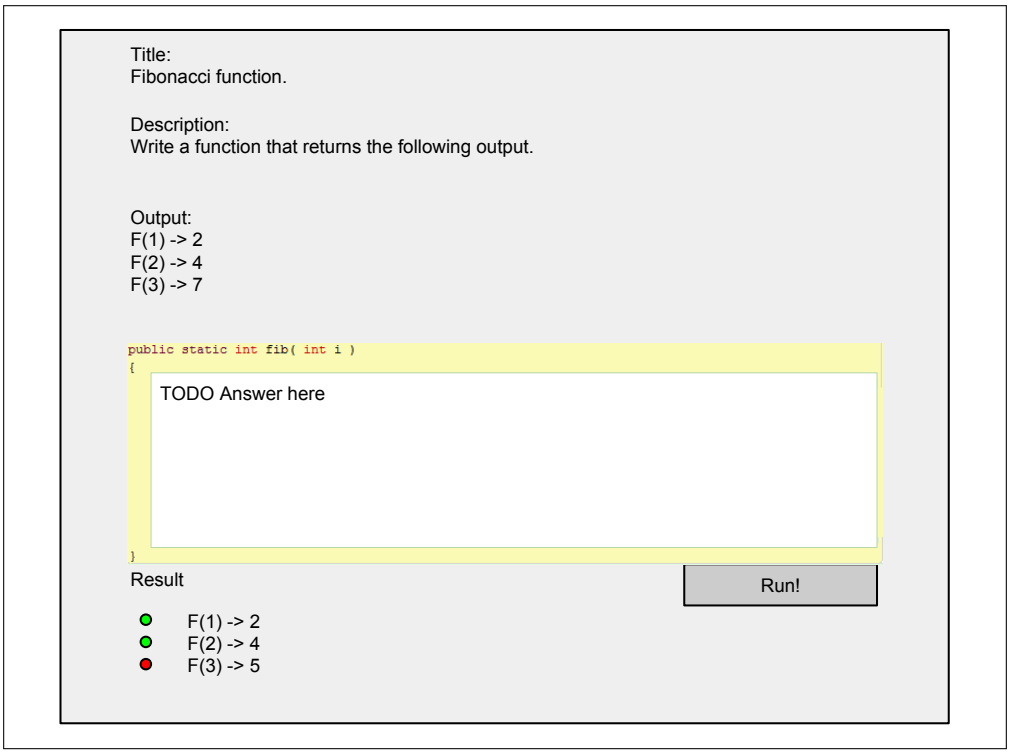
Description Design proposal containing possible workflow between the main extension and the JUnitRunner module. The diagram also includes class cards and detailed outlines of method calls, returns and read/write access between modules. The diagram helped the team understand how the JUnit runner would 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.

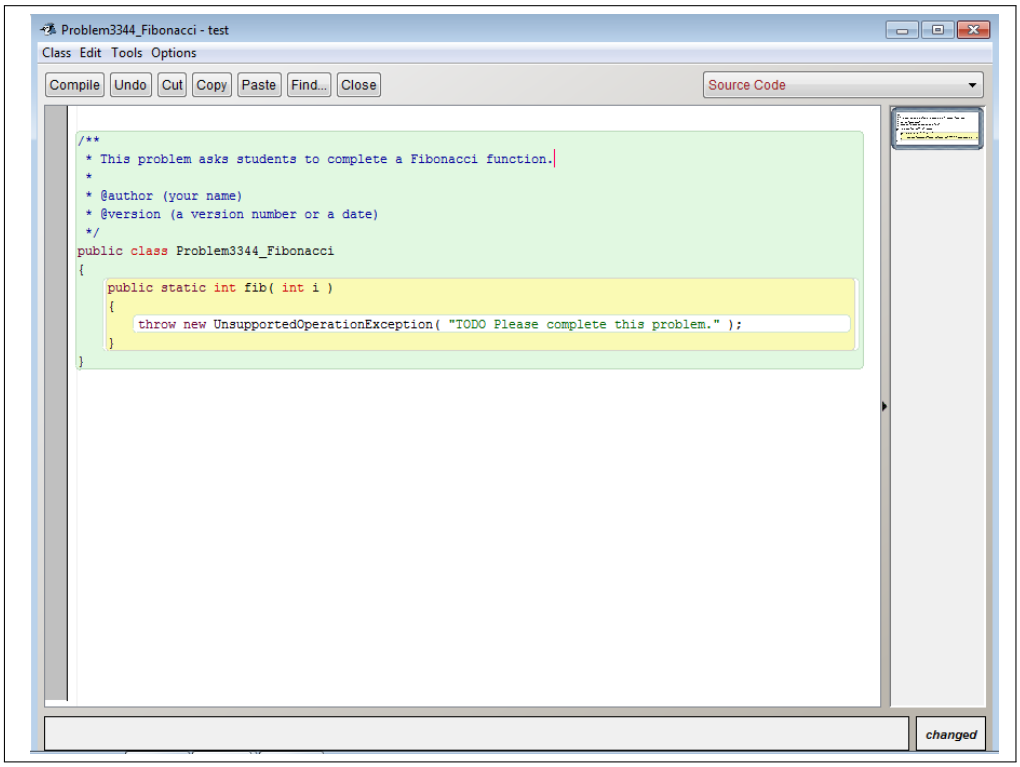


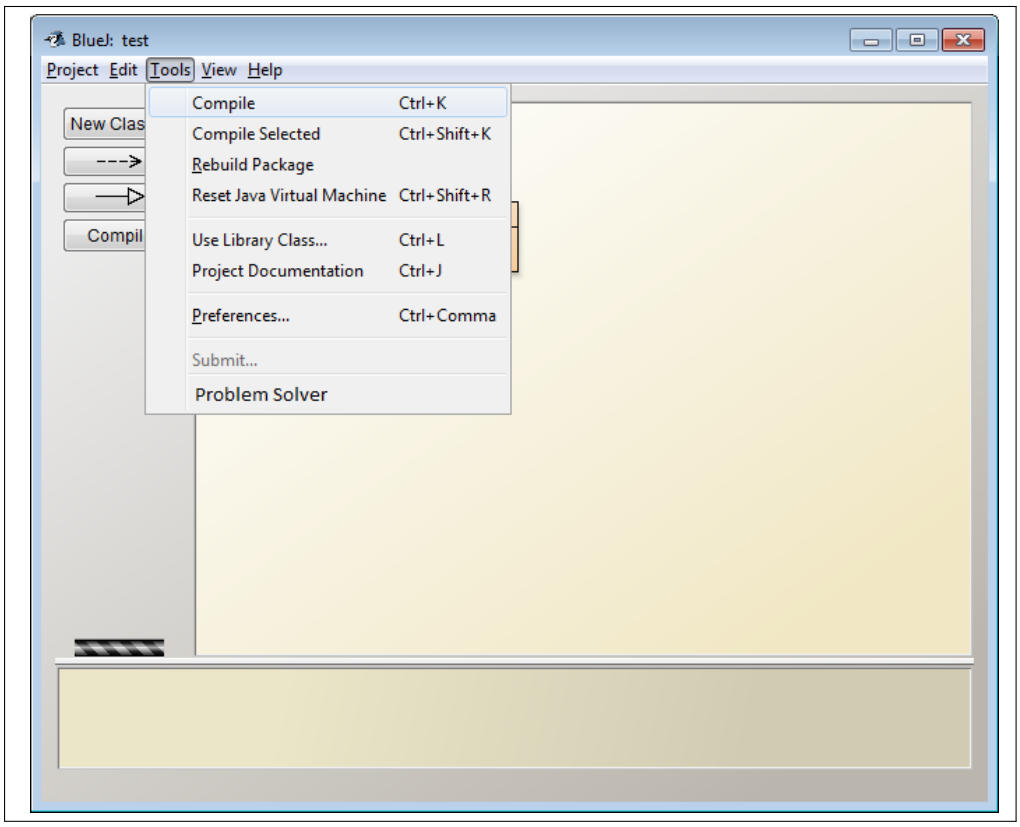
Proposed BlueJ TA Design for Graphical User Interfaces

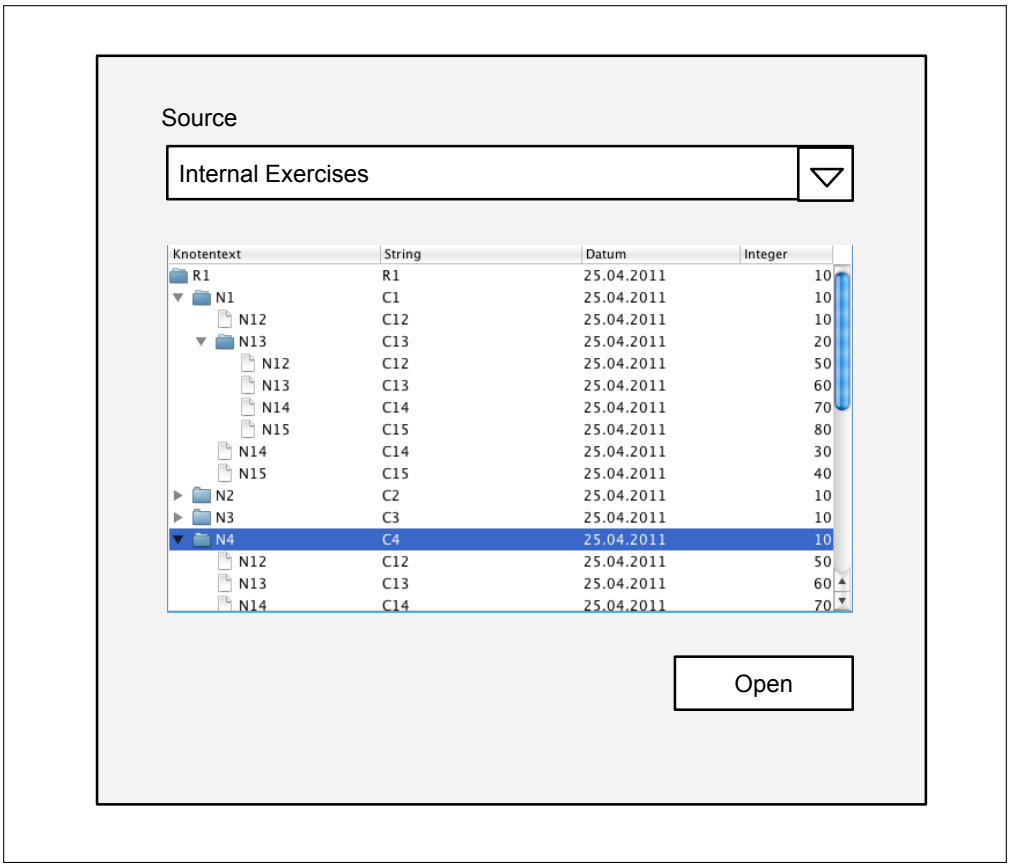
Description Our team spent time at the board designing the GUI. We created this design to cover the flow of the extension, how it would look, and functionality. These designs also helped us all have a unified idea of what the project was going to be at the end. The following artifact is what we created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


Problem3344_Fibonacci - test
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
/**
* This problem asks students to complete a Fibonacci function.
*
* @author (your name)
* @version (a version number or a date)
*/
public class Problem3344_Fibonacci
{
public static int fib(int i)
{
throw new UnsupportedOperationException("TODO Please complete this problem.");
}
}
changed


BlueJ: test
Project Edit Tools View Help
New Clas
Compil
Compile Ctrl+K
Compile Selected Ctrl+Shift+K
Rebuild Package
Reset Java Virtual Machine Ctrl+Shift+R
Use Library Class... Ctrl+L
Project Documentation Ctrl+J
Preferences... Ctrl+Comma
Submit...
Problem Solver

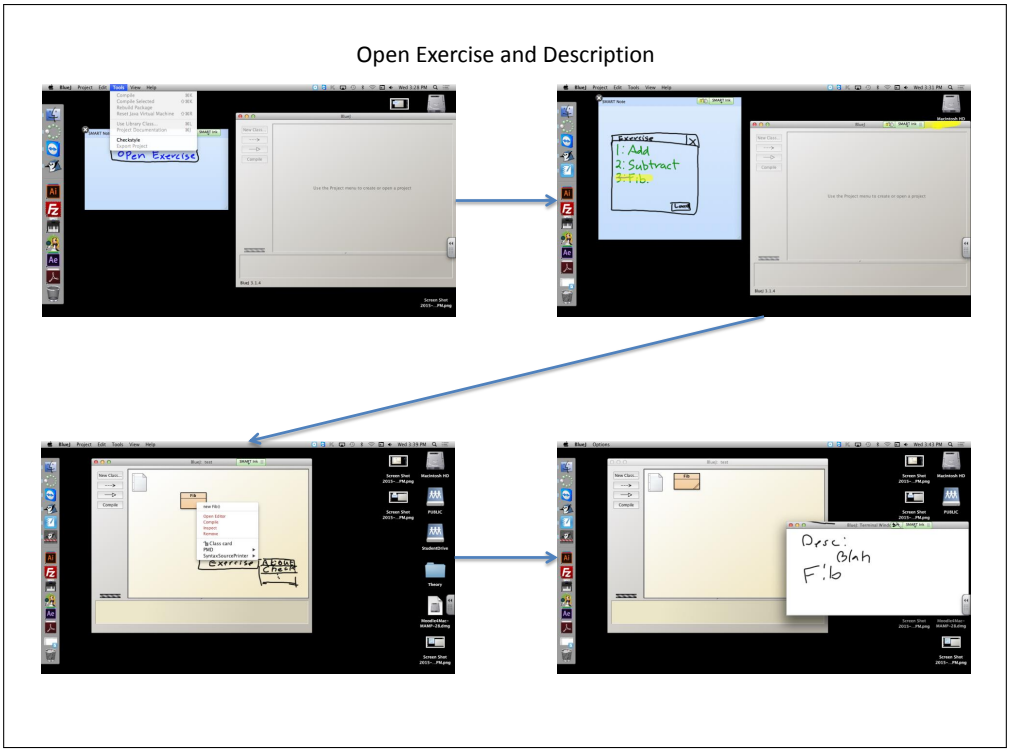

Source
Internal Exercises
Knotentext
String
Datum
Integer
R1 R1 25.04.2011 10
N1 C1 25.04.2011 10
N12 C12 25.04.2011 10
N13 C13 25.04.2011 20
N12 C12 25.04.2011 50
N13 C13 25.04.2011 60
N14 C14 25.04.2011 70
N15 C15 25.04.2011 80
N14 C14 25.04.2011 30
N15 C15 25.04.2011 40
N2 C2 25.04.2011 10
N3 C3 25.04.2011 10
N4 C4 25.04.2011 10
N12 C12 25.04.2011 50
N13 C13 25.04.2011 60
N14 C14 25.04.2011 70
Open

Note: 3 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.

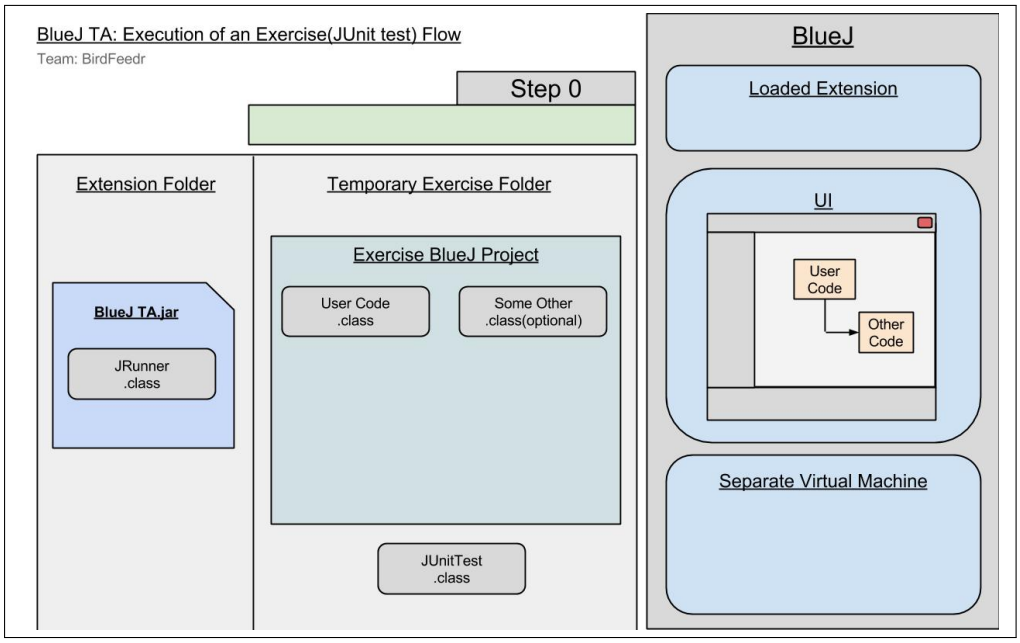
The following artifacts consists of the diagrams that describe the flow of the user in BlueJ TA.

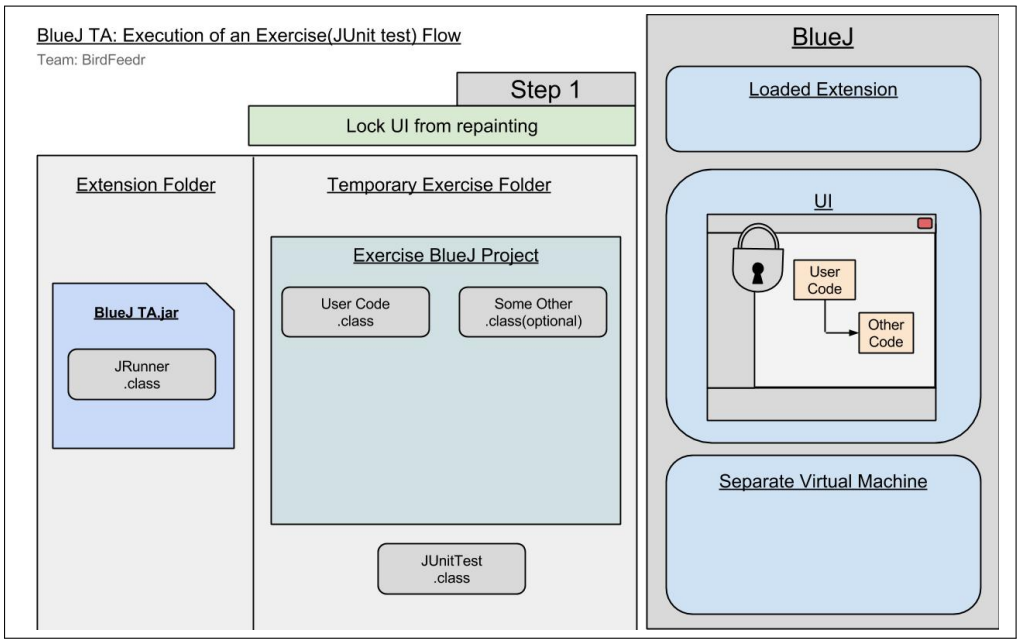


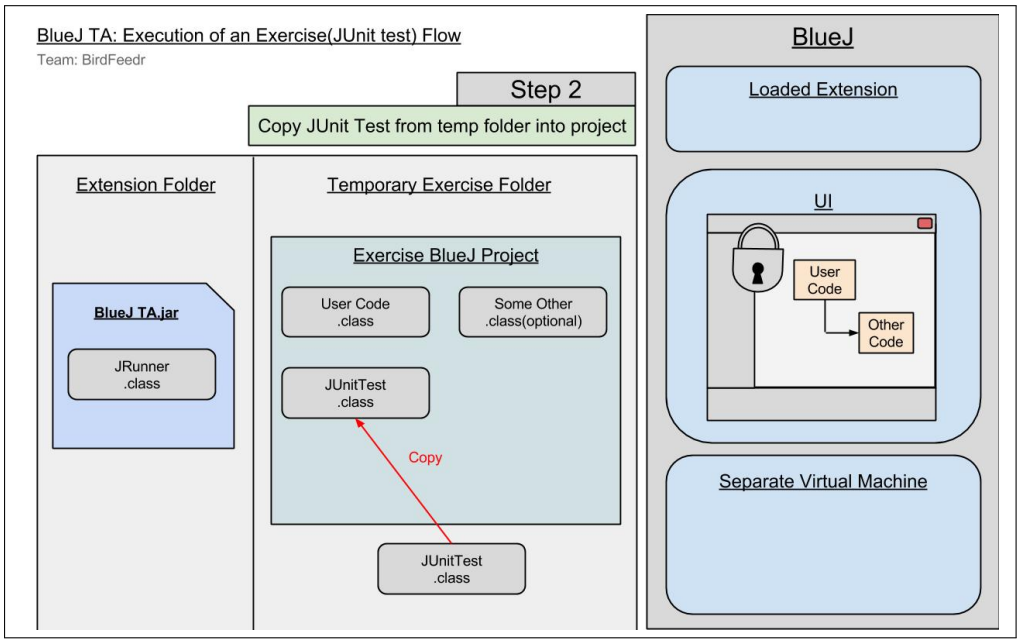
BlueJ TA Execution Design

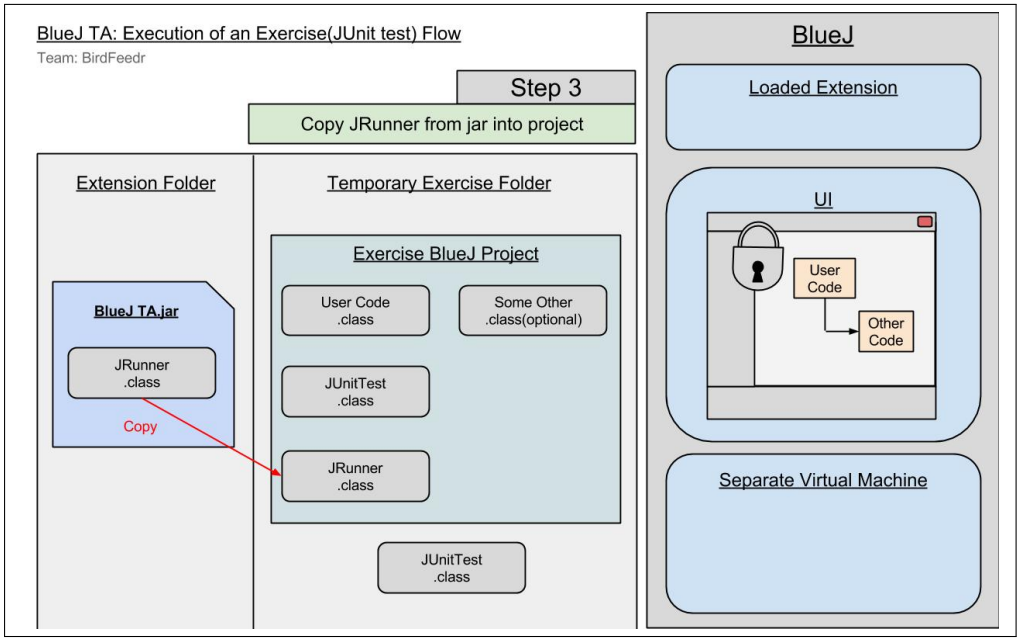
Description These designs where made to help better describe how a exercise would be executed in BlueJ TA. Showing how the files and BlueJ's virtual machine is utilized to execute the test.

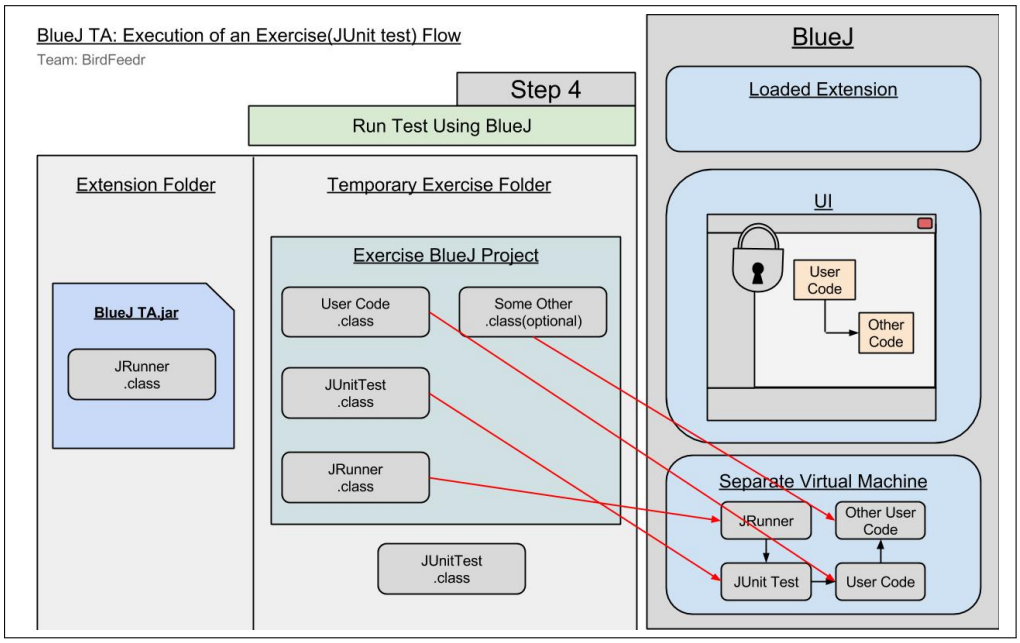
The following artifacts consists of the diagrams that describe the execution of an exercise.

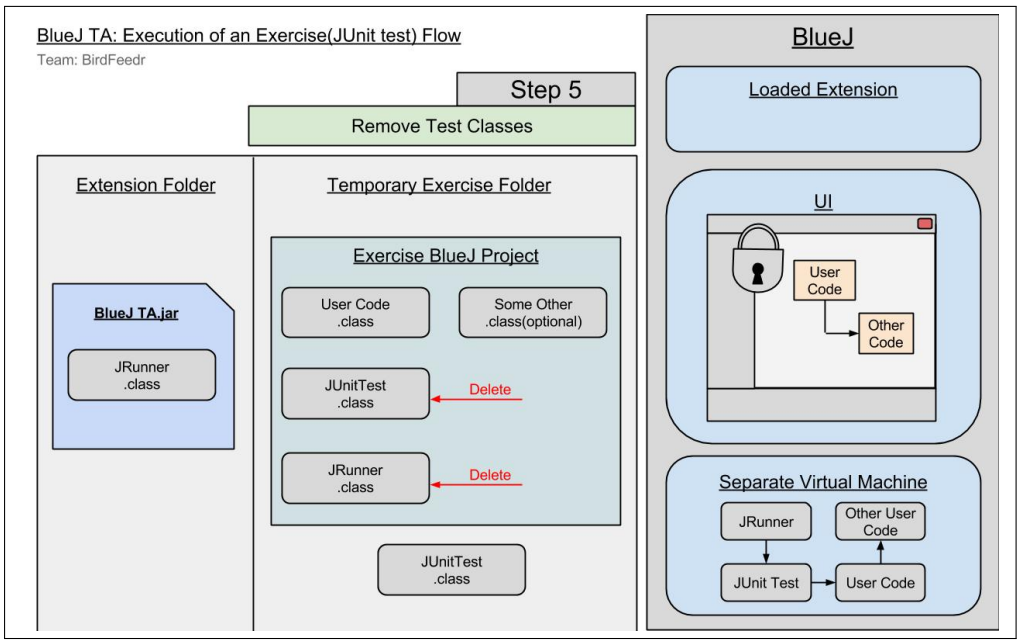


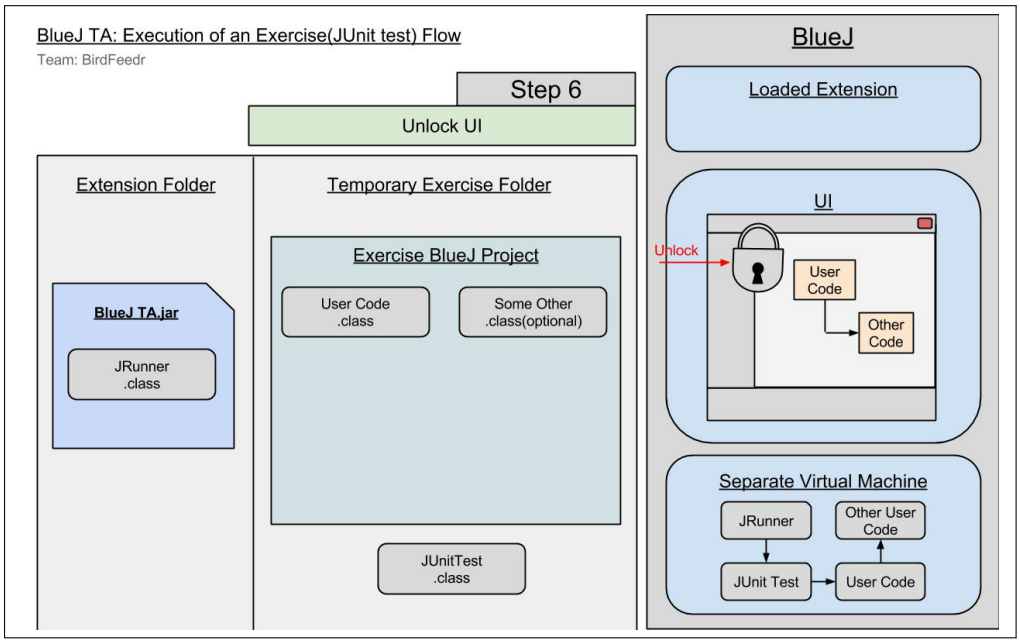


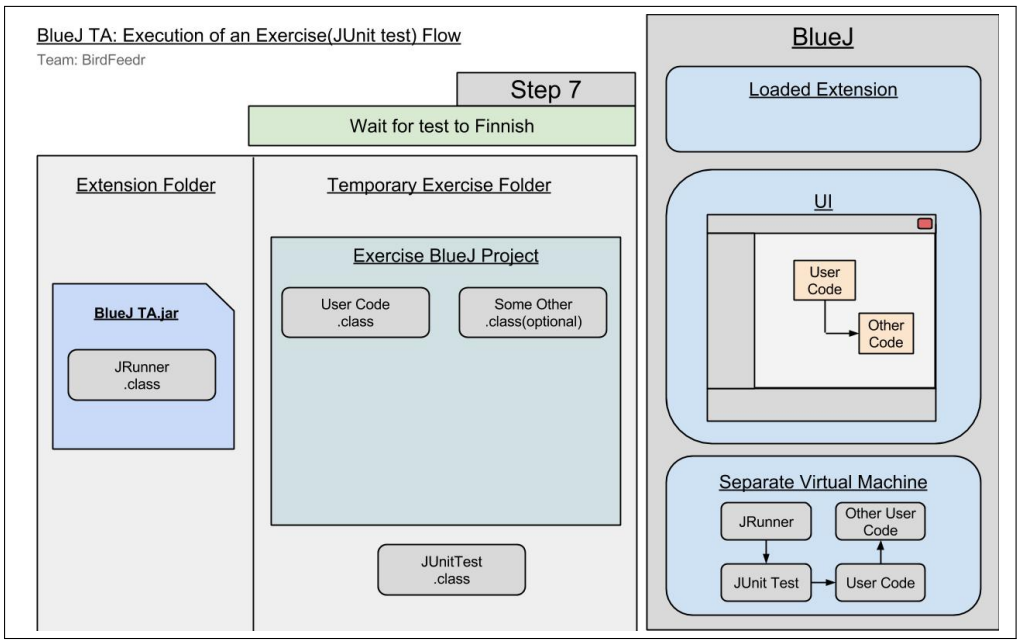

BlueJ TA: Execution of an Exercise(JUnit test) Flow
Team: BirdFeedr
Step 2
Copy JUnit Test from temp folder into project
Extension Folder
Temporary Exercise Folder
BlueJ TA.jar
JRunner .class
Exercise BlueJ Project
User Code .class
Some Other .class(optional)
JUnitTest .class
Copy
JUnitTest .class
BlueJ
Loaded Extension
UI
User Code
Other Code
Separate Virtual Machine

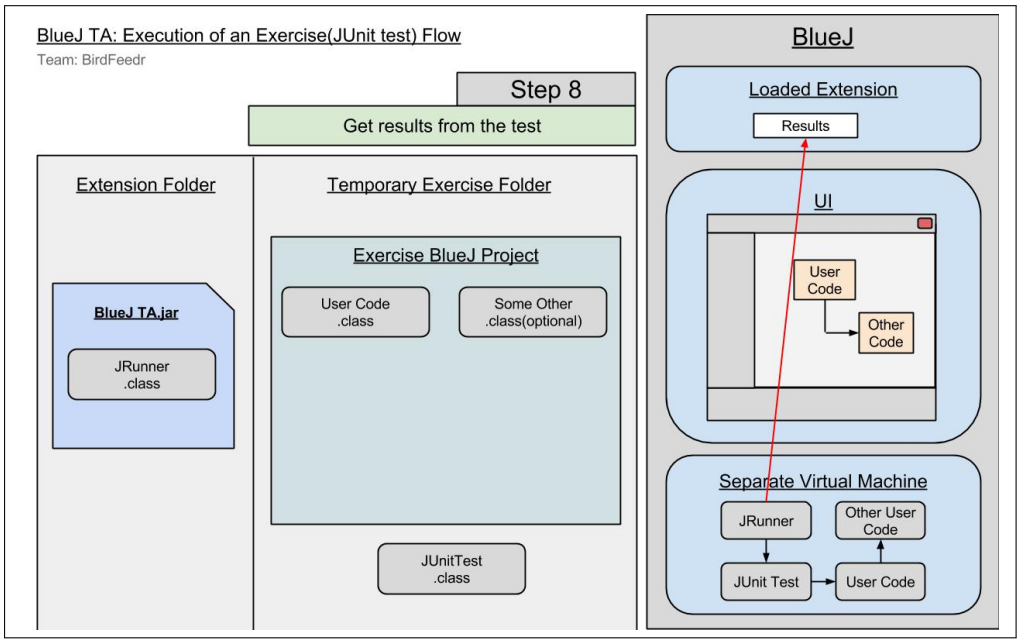

BlueJ TA: Execution of an Exercise(JUnit test) Flow
Team: BirdFeedr
Step 3
Copy JRunner from jar into project
Extension Folder
Temporary Exercise Folder
BlueJ TA.jar
JRunner .class
Copy
Exercise BlueJ Project
User Code .class
Some Other .class(optional)
JUnitTest .class
JRunner .class
JUnitTest .class
BlueJ
Loaded Extension
UI
User Code
Other Code
Separate Virtual Machine

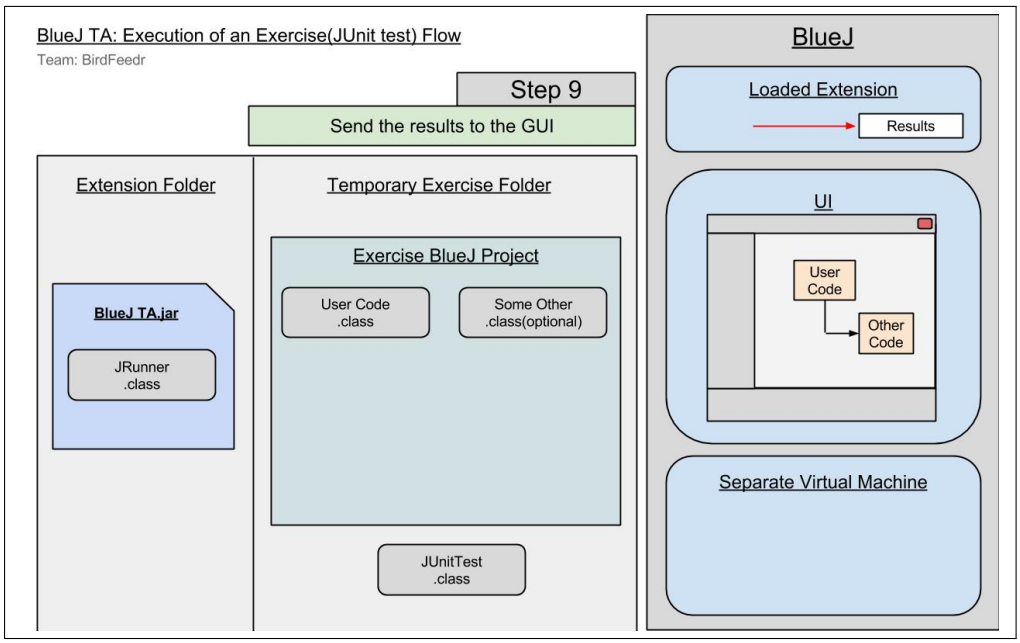

BlueJ TA: Execution of an Exercise(JUnit test) Flow
Team: BirdFeedr
Step 4
Run Test Using BlueJ
Extension Folder
Temporary Exercise Folder
BlueJ TA.jar
JRunner .class
Exercise BlueJ Project
User Code .class
Some Other .class(optional)
JUnitTest .class
JRunner .class
JUnitTest .class
BlueJ
Loaded Extension
UI
User Code
Other Code
Separate Virtual Machine
JRunner
Other User Code
JUnit Test
User Code


BlueJ TA: Execution of an Exercise(JUnit test) Flow
Team: BirdFeedr
Step 5
Remove Test Classes
Extension Folder
Temporary Exercise Folder
BlueJ TA.jar
JRunner .class
Exercise BlueJ Project
User Code .class
Some Other .class(optional)
JUnitTest .class
Delete
JRunner .class
Delete
JUnitTest .class
BlueJ
Loaded Extension
UI
User Code
Other Code
Separate Virtual Machine
JRunner
Other User Code
JUnit Test
User Code


BlueJ TA: Execution of an Exercise(JUnit test) Flow
Team: BirdFeedr
Step 6
Unlock UI
Extension Folder
Temporary Exercise Folder
BlueJ TA.jar
JRunner .class
Exercise BlueJ Project
User Code .class
Some Other .class(optional)
JUnitTest .class
BlueJ
Loaded Extension
UI
Unlock
User Code
Other Code
Separate Virtual Machine
JRunner
Other User Code
JUnit Test
User Code


BlueJ TA: Execution of an Exercise(JUnit test) Flow
Team: BirdFeedr
Step 7
Wait for test to Finnish
Extension Folder
Temporary Exercise Folder
BlueJ TA.jar
JRunner .class
Exercise BlueJ Project
User Code .class
Some Other .class(optional)
JUnitTest .class
BlueJ
Loaded Extension
UI
User Code
Other Code
Separate Virtual Machine
JRunner
Other User Code
JUnit Test
User Code


BlueJ TA: Execution of an Exercise(JUnit test) Flow
Team: BirdFeedr
Step 8
Get results from the test
Extension Folder
Temporary Exercise Folder
Exercise BlueJ Project
BlueJ TA.jar
JRunner .class
User Code .class
Some Other .class(optional)
JUnitTest .class
BlueJ
Loaded Extension
Results
UI
User Code
Other Code
Separate Virtual Machine
JRunner
Other User Code
JUnit Test
User Code


BlueJ TA: Execution of an Exercise(JUnit test) Flow
Team: BirdFeedr
Step 9
Send the results to the GUI
Extension Folder
Temporary Exercise Folder
Exercise BlueJ Project
BlueJ TA.jar
JRunner .class
User Code .class
Some Other .class(optional)
JUnitTest .class
BlueJ
Loaded Extension
Results
UI
User Code
Other Code
Separate Virtual Machine

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.

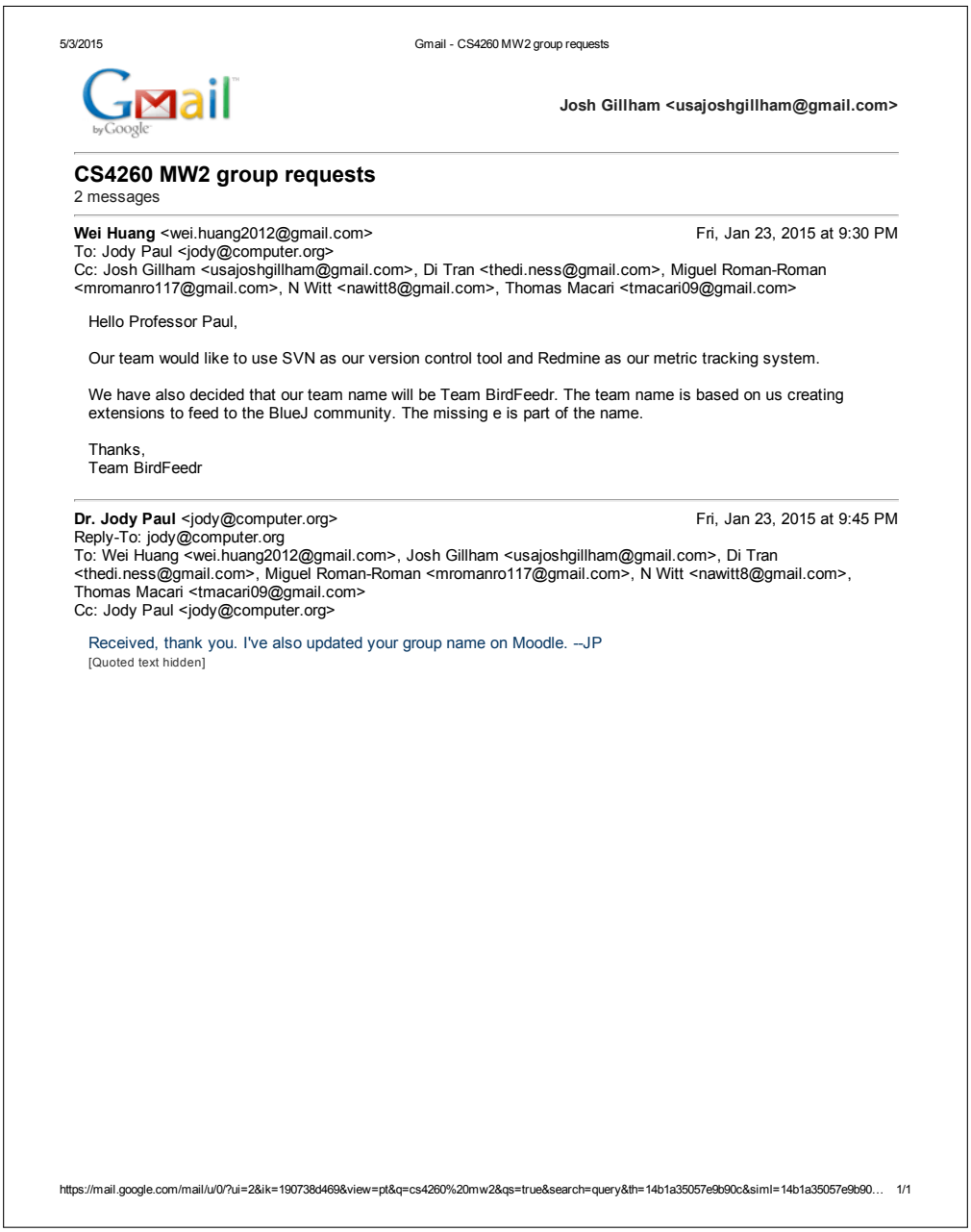
```
BEGIN File: /Team Documentation/Meeting Notes/note-20150123Team-Meeting.txt
```

```
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.


Wei: Understands what we mean by the project we want to do.


Wei: Puts his vote in for svn b/c it comes with redmine.

Di: votes the same.

Miguel: Votes the same.

Nate: says its the one he used to.

Josh: Since we know those tools we don't have to spend time learning.

Thomas: has never used redmine.

Wei: YOu just need to knw how to write wiki pages and create issues.


Team names Ideas
Bird feeder
Extendables


Characteristics:
```

```
Wei: Proud of it.
Nathan: Something that motivates us.
Di: BirdFeedr


We are choosing Team BirdFeedr w/o the last e because we are making extensions.
 We invision our team as feeding the BlueJ community.

---

Wei: Wants one out of class meeting per week.
 Make sure everyone is on the same page.
 We can do it more often as needed.
 A few key people need to be there.
 i.e. only people that are coding the extension.

JOsh: one or more meetings weekly.

Di: The weekly the meeting will allow us to touch base.

Wei: class will be mostly teaching.
  Record each meeting so we can keep track.

Wei: we could have a folder in svn for reports corresponding to the project
  notebook.

Thomas: volunteering for managing svn.

  Individuals do not have a notebook but rather a portfolio.

Wei: Update the project notebook after every meeting.

Thomas: The notebook should be in the svn repo.

Josh: There will be a report after each meeting in the notebook.

Wei: Current status -- deciding our extension.
  Everyone should report on that on Monday.

Nathan: Discuss the extension on monday. Clearify.
Miguel Research

FInished svn, metric tracking, team name.
Todo Process model, choosing extension.

Josh: exp for tiem tracking.

Nathan: Suggestion -- Note what you did b4 the day is over.

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



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

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"

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

Code
+---build.xml                        The build file compiles, installs, and
|                                     runs the extension.
+---local.properties.macOSX          Specifies build properties for a Mac.
+---local.properties.unix            Specifies build properties for a Linux.
+---local.properties.windows         Specifies build properties for a Windows.
\---src                              The source code of the project.
                                     Note: this is itself a BlueJ project.


================================================================================
Manual build instructions:

Warning!
Please use the ant build file as it builds everything for you!
Manual building should be done only if the ant build file fails.

Note: Replace [and anything inside]
1. Create manifest.txt with following text inside:
Main-Class: [Your startup file. No file extension]
2. Move manifest.txt to your extension's .class files
3. Open command prompt and navigate to your .class files
4. Type in the follow command prompt argument to build:
jar cmf manifest.txt [Name of jar file].jar *.class
5. The Bluej extension .jar file should be in your current directory.
```

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

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

Installing an extension into BlueJ:

Drop the extension jar file into the extension folder for bluej to install.
Instructions on creating the jar file above.

For all users of this system in all projects.

<BLUEJ_HOME>/lib/extensions (Unix),
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.
         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


BEGIN File: /README.txt
```

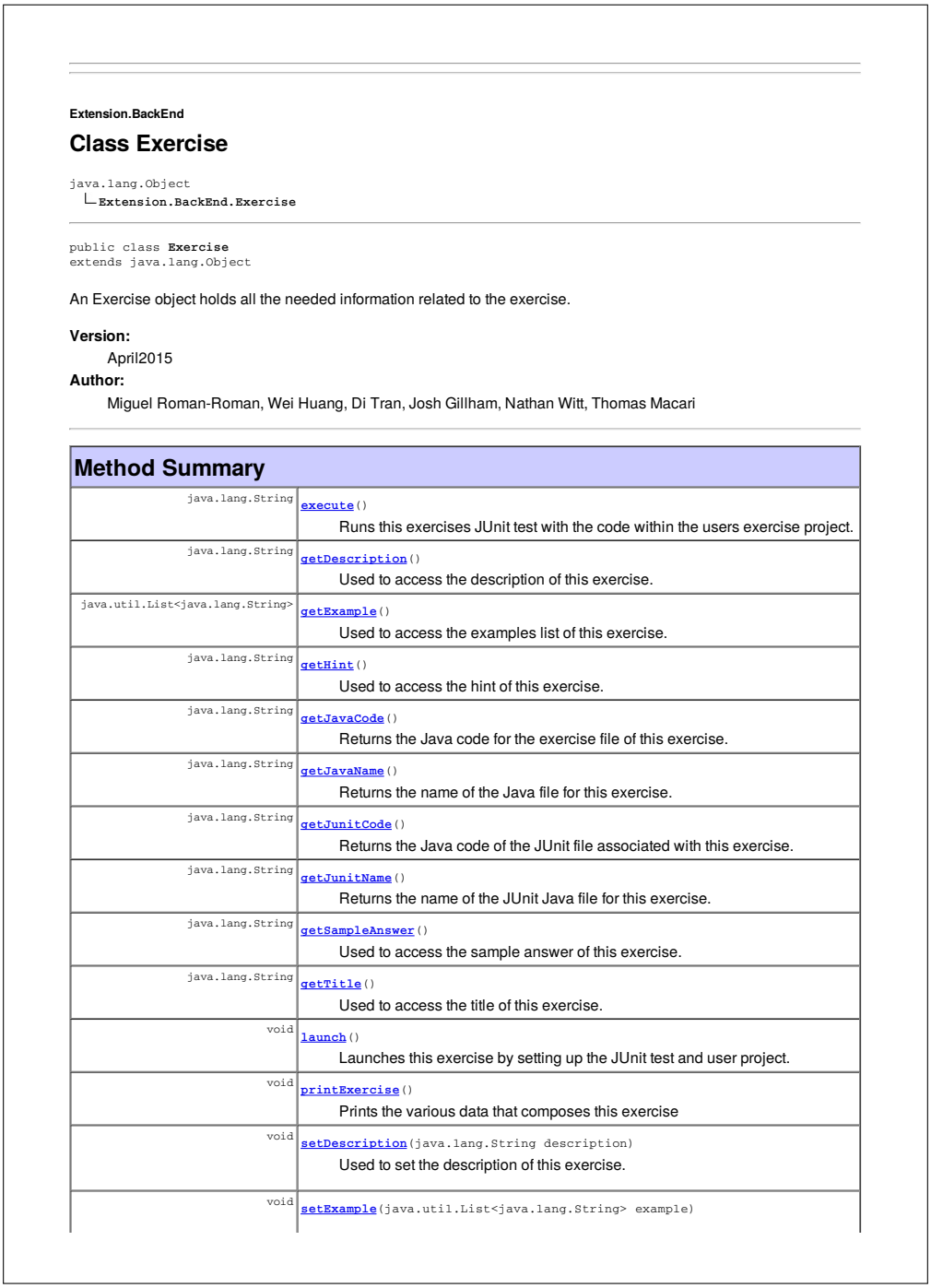
```
Last Edited: 5-1-2015 at 4:59 PM
Current active project: BlueJ TA

Location of Java source files: \Projects\BlueJ_TA\Code\src
Location of ant build file:    \Projects\BlueJ_TA\Code\build.xml
Location of README:            \Projects\BlueJ_TA\Code\README.txt

Currently the best way to create the jar for Project BlueJ TA is to use
the ant build file above.
For more information about the project or manual build instructions see the
README.txt as above.

================================================================================
Structure of repository:

Location:                              Explanation:
--------------------------------------------------------------------------------
|birdfeedr
+---Projects                           -- All Projects.
|   +---BlueJ_TA                       -- Project BlueJ TA.
|   |   +---Code                       -- The heart of the project.
|   |   |   \---README.txt             -- Explains how extension works.
|   |   \---Documentation              -- Design and management artifacts.
|   |       +---Designs                -- GUI and internal designs (outdated).
|   |       |   \---Do Function        -- GUI flow design (outdated).
|   |       +---Possible Templates     -- Java entity ideas.
|   |       +---Process Model          -- Failed attempt to formalize process.
|   |       +---Survey                 -- The student/teacher survey.
|   |       \---Tour                   -- A collection of screen shots.
|   +---Chirp                          -- Folder for Project Chirp.
|   |   +---Dist                       -- Distribution of Project Chirp.
|   |   +---src                        -- Java source files for project Chirp.
|   |   \---Exploration Summary.txt    -- Experience gained from Project Chirp.
|   \---Git                            -- Folder for Project Git.
|       +---Documents                  -- Project Documentation.
|       \---Git-Extension              -- source files for Project Git.
+---Sandbox                            -- Individual team member folders.
|   +---README.txt                         -- A description of the Sandbox.
|   +---(Group member's name)*         -- Folder for individual member's use.
\---Team Documentation                 -- All Team decumentation.
    +---Meeting Notes                  -- Notes from group meetings. (informal)
    +---Presentations                  -- All presentations.
    +---Reports                        -- All team reports.
    +---Team Portfolio                 -- The project Notebook.
    \---Weekly Summary                 -- Weekly status updates. (formal)
    \---Birdfeedr Policies.txt         -- A text file for group policies.
```

```
END File: /README.txt
```

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

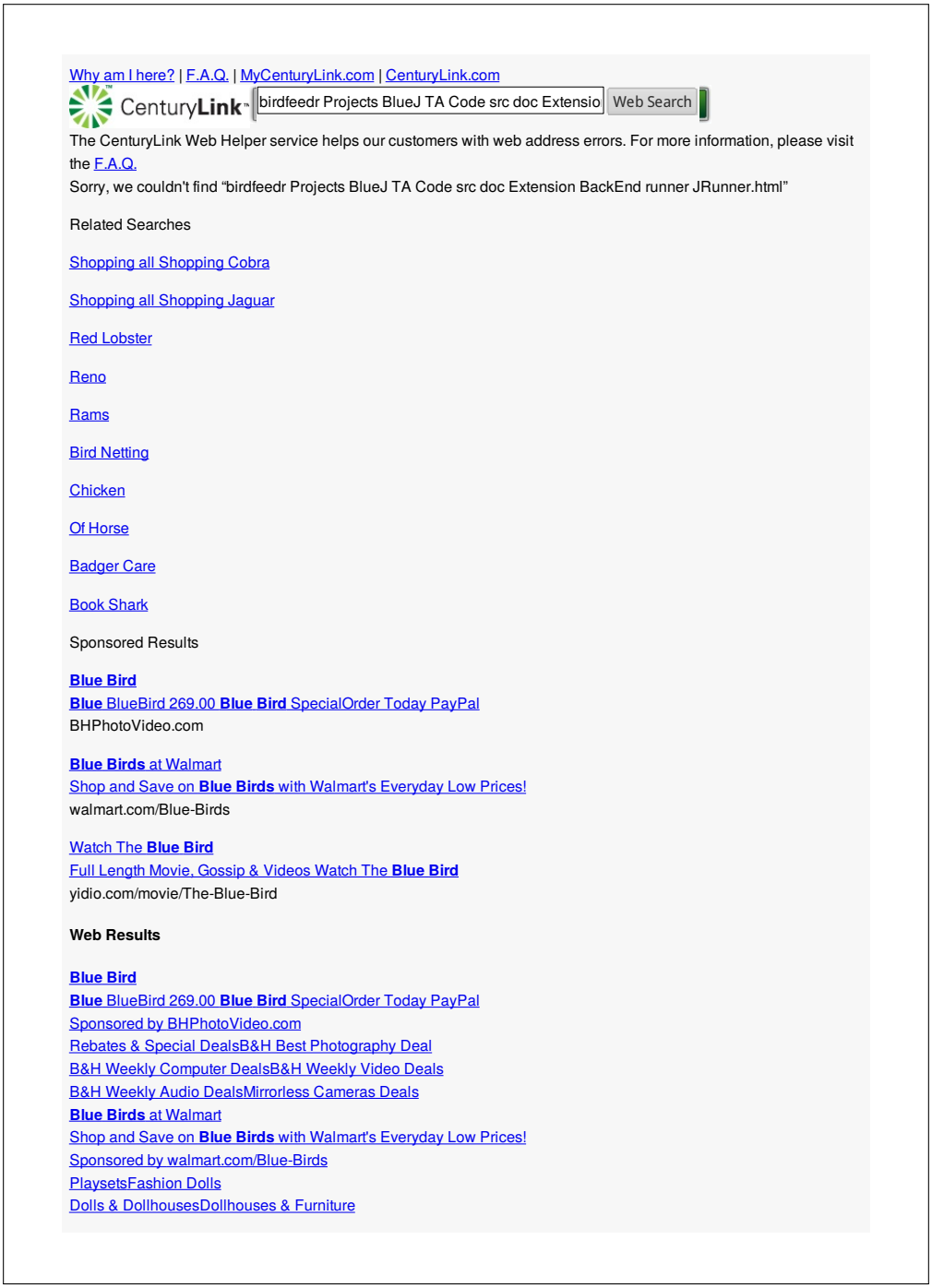
RemoveThread(bluej.extensions.BClass testClass, bluej.extensions.BClass runClass, java.awt.Frame frame) Constructor

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


Why am I here? | F.A.Q. | MyCenturyLink.com | CenturyLink.com
CenturyLink
birdfeedr Projects BlueJ TA Code src doc Extensio
Web Search
The CenturyLink Web Helper service helps our customers with web address errors. For more information, please visit the F.A.Q.
Sorry, we couldn't find "birdfeedr Projects BlueJ TA Code src doc Extension BackEnd runner JRunner.html"
Related Searches
Shopping all Shopping Cobra
Shopping all Shopping Jaguar
Red Lobster
Reno
Rams
Bird Netting
Chicken
Of Horse
Badger Care
Book Shark
Sponsored Results
Blue Bird
Blue BlueBird 269.00 Blue Bird SpecialOrder Today PayPal
BHPhotoVideo.com
Blue Birds at Walmart
Shop and Save on Blue Birds with Walmart's Everyday Low Prices!
walmart.com/Blue-Birds
Watch The Blue Bird
Full Length Movie, Gossip & Videos Watch The Blue Bird
yidio.com/movie/The-Blue-Bird
Web Results
Blue Bird
Blue BlueBird 269.00 Blue Bird SpecialOrder Today PayPal
Sponsored by BHPhotoVideo.com
Rebates & Special DealsB&H Best Photography Deal
B&H Weekly Computer DealsB&H Weekly Video Deals
B&H Weekly Audio DealsMirrorless Cameras Deals
Blue Birds at Walmart
Shop and Save on Blue Birds with Walmart's Everyday Low Prices!
Sponsored by walmart.com/Blue-Birds
PlaysetsFashion Dolls
Dolls & DollhousesDollhouses & Furniture

Extension.BackEnd

Class StateManager

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

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

The StateManager class controls the current state of the extension. It holds all necessary information such as the GUI's and the currently chosen exercise.

Version:
April2015

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

Method Summary

static void	createDescriptionGUI() Creates the GUI for the description of the exercise.
static void	createExerciseListGUI() Creates the GUI for picking an exercise.
static void	createTestResultsGUI() Creates the GUI showing the JUnit tests for the exercise.
static void	disposeDescriptionGUI() Disposes the DescriptionGUI
static void	disposeExerciseGUI() Disposes the ExerciseListGUI
static void	disposeTestResultsGUI() Disposes the TestResultsGUI
static bluej.extensions.BlueJ	getBlueJ() Returns the BlueJ instance held
static Extension.BackEnd.Exercise	getCurrentExercise() Returns the current exercise chosen by the user.
static Extension.GUI.DescriptionGUI	getDescriptionGUI() Returns the held instance of the DescriptionGUI
static java.util.ArrayList<Extension.BackEnd.Exercise>	getExerciseList() Initializes the exercise list.
static Extension.GUI.ExerciseListGUI	getExerciseListGUI() Returns the held instance of the ExerciseListGUI
static java.lang.String	getExercisesLocation() Returns the String location of the exercises.
static java.io.File	getTempDir() Returns a File for the location of the Temporary Directory created by this extension.

Extension.BackEnd

Class XMLReader

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

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

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

Version:

April2015

Author:

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

Constructor Summary

XMLReader()   Constructor

Method Summary

Extension.BackEnd.Exercise	readExercise(java.io.InputStream is) Reads an XML file that has been converted into an InputStream
Extension.BackEnd.Exercise	readExercise(java.lang.String filePath) Reads in an XML file from the given path

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

Constructor Detail

XMLReader

```
public XMLReader()
```

Constructor

Method Detail

readExercise

```
public Extension.BackEnd.Exercise readExercise(java.io.InputStream is)
```

Reads an XML file that has been converted into an InputStream

Parameters:

Extension.GUI

Class DescriptionGUI

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
                      └Extension.GUI.DescriptionGUI
```

All Implemented Interfaces:

java.awt.event.WindowListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.util.EventListener, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

---

```
public class DescriptionGUI
extends javax.swing.JFrame
implements java.awt.event.WindowListener
```

The Graphical User Interface used to show the description of the exercises the user is working on.

Version:

April2015

Author:

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

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Nested classes/interfaces inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Nested classes/interfaces inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow, java.awt.Window.Type

Nested classes/interfaces inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes/interfaces inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Extension.GUI

Class ExerciseListGUI

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
                      └Extension.GUI.ExerciseListGUI
```

All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

---

```
public class ExerciseListGUI
extends javax.swing.JFrame
```

The Graphical User Interface used to show the user the list of exercises along with the description and allows them to load an exercise

Version:

April2015

Author:

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

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Nested classes/interfaces inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Nested classes/interfaces inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow, java.awt.Window.Type

Nested classes/interfaces inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes/interfaces inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Extension.GUI

Class FXMLDescriptionDocumentController

```
java.lang.Object
  └ Extension.GUI.FXMLDescriptionDocumentController
```

All Implemented Interfaces:

javafx.fxml.Initializable

```
public class FXMLDescriptionDocumentController
extends java.lang.Object
implements javafx.fxml.Initializable
```

Used by the FXML document to control the description GUI

Version:

April2015

Author:

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

Constructor Summary

FXMLDescriptionDocumentController()

Method Summary

void	initialize(java.net.URL url, java.util.ResourceBundle rb) Initializes this GUI with the proper data

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

Constructor Detail

FXMLDescriptionDocumentController

```
public FXMLDescriptionDocumentController()
```

Method Detail

initialize

```
public void initialize(java.net.URL url,
                       java.util.ResourceBundle rb)
```

Initializes this GUI with the proper data

Specified by:

initialize in interface javafx.fxml.Initializable

Extension.GUI

Class FXMLExerciseDocumentController

```
java.lang.Object
  └ Extension.GUI.FXMLExerciseDocumentController
```

All Implemented Interfaces:

javafx.fxml.Initializable

```
public class FXMLExerciseDocumentController
extends java.lang.Object
implements javafx.fxml.Initializable
```

Used by the FXML document to control the ExerciseList GUI.

Version:

April2015

Author:

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

Constructor Summary

FXMLExerciseDocumentController()

Method Summary

void	initialize(java.net.URL url, java.util.ResourceBundle rb) Used to initialize the GUI

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

Constructor Detail

FXMLExerciseDocumentController

```
public FXMLExerciseDocumentController()
```

Method Detail

initialize

```
public void initialize(java.net.URL url,
                       java.util.ResourceBundle rb)
```

Used to initialize the GUI

Specified by:

`initialize` in interface `javafx.fxml.Initializable`

Extension.GUI

Class FXMLTestResultsDocumentController

```
java.lang.Object
  └ Extension.GUI.FXMLTestResultsDocumentController
```

All Implemented Interfaces:

javafx.fxml.Initializable

```
public class FXMLTestResultsDocumentController
extends java.lang.Object
implements javafx.fxml.Initializable
```

Used by the FXML document to control the test results GUI

Version:

April2015

Author:

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

Constructor Summary

FXMLTestResultsDocumentController()

Method Summary

void	initialize(java.net.URL url, java.util.ResourceBundle rb) Used to initialize this GUI
void	runTest() Gets the results from running the JUnit tests then formats the failures and prints it to the text area.

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

Constructor Detail

FXMLTestResultsDocumentController

```
public FXMLTestResultsDocumentController()
```

Method Detail

initialize

```
public void initialize(java.net.URL url,
                       java.util.ResourceBundle rb)
```

Used to initialize this GUI

Extension.GUI

Class TestResultsGUI

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
                      └Extension.GUI.TestResultsGUI
```

All Implemented Interfaces:

java.awt.event.WindowListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.util.EventListener, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

---

```
public class TestResultsGUI
extends javax.swing.JFrame
implements java.awt.event.WindowListener
```

The Graphical User Interface used to display test result data to the user after the JUnit tests have been run on the user's code.

Version:

April2015

Author:

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

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Nested classes/interfaces inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Nested classes/interfaces inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow, java.awt.Window.Type

Nested classes/interfaces inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes/interfaces inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Extension

Class Main

```
java.lang.Object
  └bluej.extensions.Extension
      └Extension.Main
```

```
public class Main
extends bluej.extensions.Extension
```

The base point of this BlueJ extension.

Version:

April2015

Author:

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

Field Summary

Fields inherited from class bluej.extensions.Extension
VERSION_MAJOR, VERSION_MINOR

Constructor Summary

Main()

Method Summary

java.lang.String	getName() Used to get the name of this extension
java.lang.String	getVersion() Used to get the current version of this extension
boolean	isCompatible() Used to check if this extension is compatible with the BlueJ version used
void	startup(bluej.extensions.BlueJ blueJ) Used at start up of the extension
void	terminate() Called when this extension is being terminated

Methods inherited from class bluej.extensions.Extension
getDescription, getURL

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

Extension

Class MenuBuilder

```
java.lang.Object
  └bluej.extensions.MenuGenerator
      └Extension.MenuBuilder
```

```
public class MenuBuilder
extends bluej.extensions.MenuGenerator
```

Creates menu items for this BlueJ extension

Version:

April2015

Author:

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

Constructor Summary

MenuBuilder()

Method Summary

javax.swing.JMenuItem	getClassMenuItem(bluej.extensions.BClass bClass) Adds extension menu items to the BlueJ class card
javax.swing.JMenuItem	getToolsMenuItem(bluej.extensions.BPackage bPackage) Adds extension menu items to the BlueJ tools menu

Methods inherited from class bluej.extensions.MenuGenerator

getMenuItem, getObjectMenuItem, getPackageMenuItem, getViewMenuItem, notifyPostClassMenu, notifyPostObjectMenu, notifyPostPackageMenu, notifyPostToolsMenu, notifyPostViewMenu

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MenuBuilder

```
public MenuBuilder()
```

Method Detail

getClassMenuItem

```
public javax.swing.JMenuItem getClassMenuItem(bluej.extensions.BClass bClass)
```

Adds extension menu items to the BlueJ class card

Extension

Class Preferences

```
java.lang.Object
  └ Extension.Preferences
```

All Implemented Interfaces:

bluej.extensions.PreferenceGenerator

```
public class Preferences
extends java.lang.Object
implements bluej.extensions.PreferenceGenerator
```

This class is used to generate a Panel within BlueJ Preferences->Extension

Version:

April2015

Author:

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

Field Summary

static java.lang.String	PROFILE_LABEL

Constructor Summary

Preferences(bluej.extensions.BlueJ bluej) Creates the Panel within BlueJ's Preferences

Method Summary

javax.swing.JPanel	getPanel() Returns the JPanel created by this object
void	loadValues() Used to load the previously saved path given by the user from the BlueJ properties file
void	saveValues() Used to save the path given by the user into the BlueJ properties file

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

Field Detail

PROFILE_LABEL

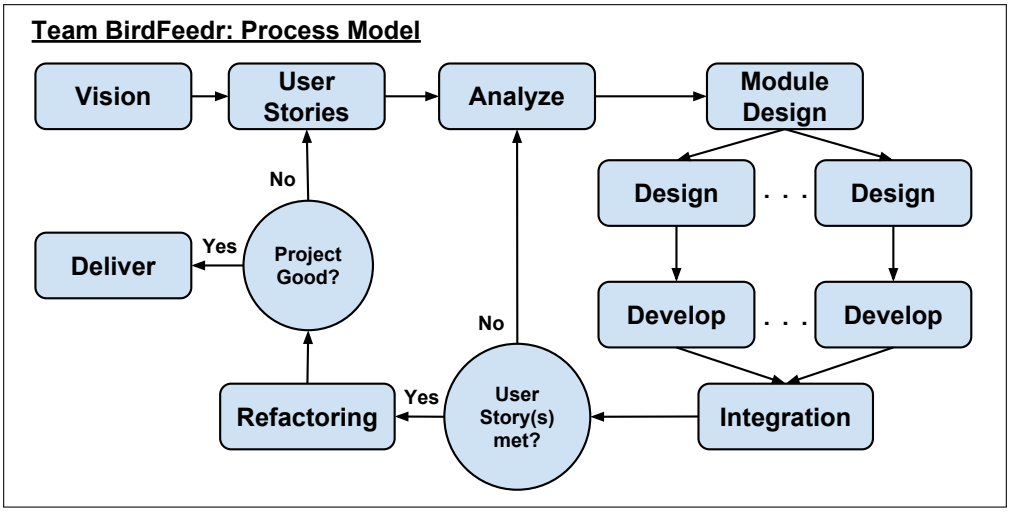
```
public static final java.lang.String PROFILE_LABEL
```

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.

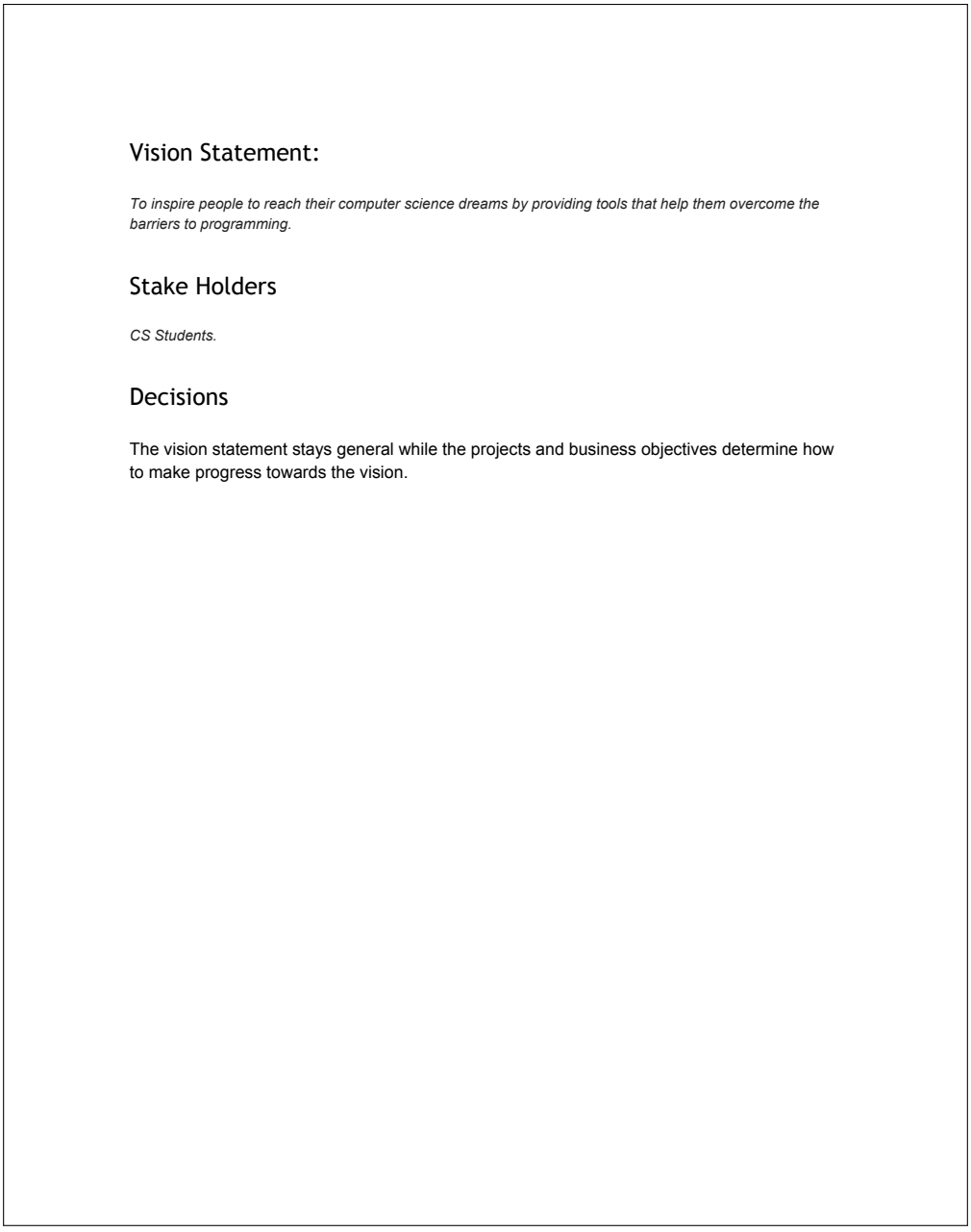
Process Model

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.

4/8/2015 Creative Commons — Attribution-NonCommercial 4.0 International — CC BY-NC 4.0

Creative Commons

Creative Commons License Deed

Attribution-NonCommercial 4.0 International (CC BY-NC 4.0)

This is a human-readable summary of (and not a substitute for) the license.
Disclaimer

You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.

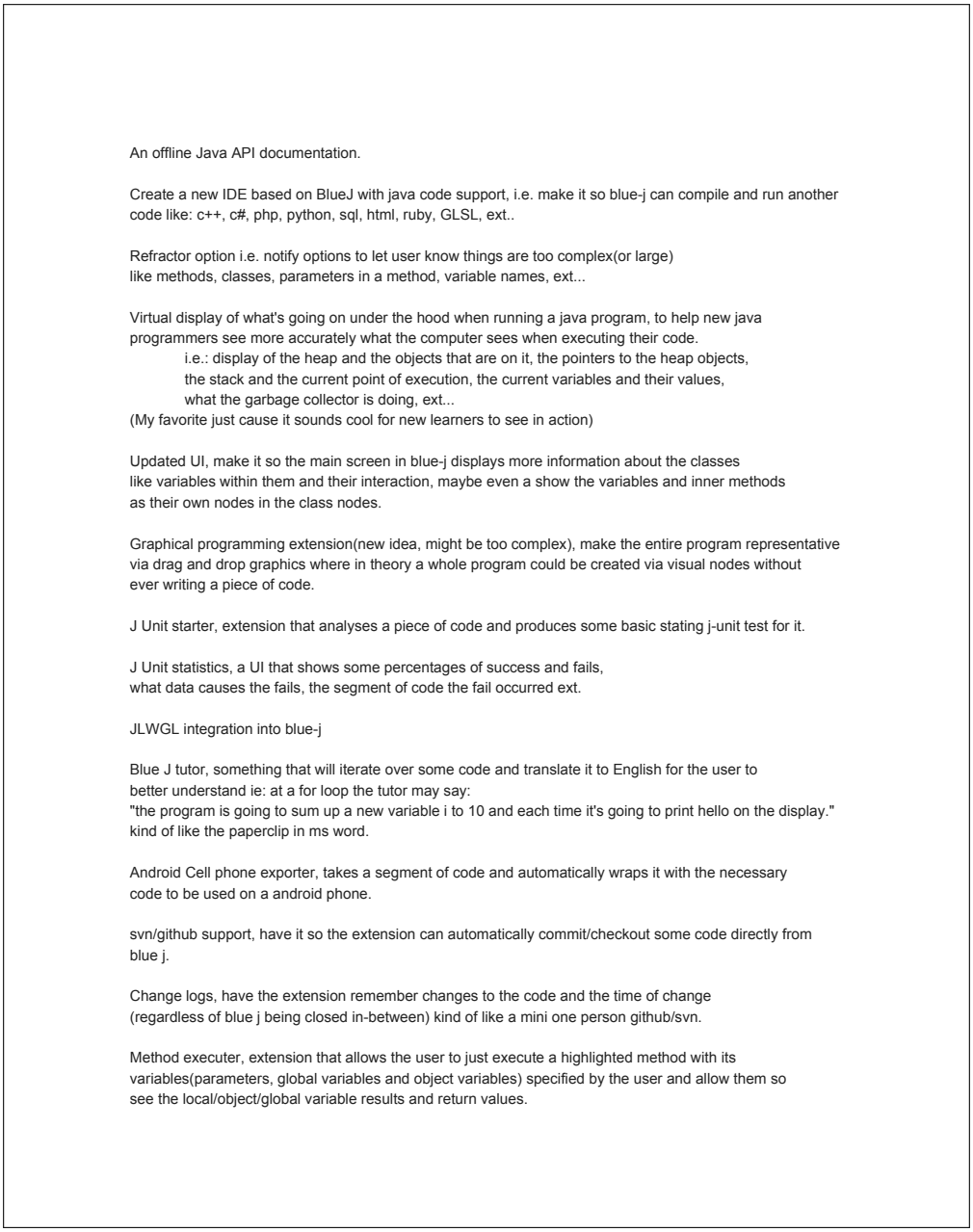
No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

The applicable mediation rules will be designated in the copyright notice published with the work, or if none

http://creativecommons.org/licenses/by-nc/4.0/ 1/2

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?
      Suggest changes to the original writer, or approve it.


2. MEETINGS
**********
CONDUCT
- - - - - - -
What is the expected level of conduct?
        Keep things civilized. No personal attacks. If conflict arises
        in a meeting, resolve it or  through mediation.
How long am I allowed to talk?
        Three minutes, unless what you have to say is really important.
I just interrupted someone, what do I do?
        Stop talking and let them continue. If they explicitly state you can
continue, then
        you may continue talking.
Do I have to go to this meeting?
        If you were invited, you have to go. Sorry.
What if I want to go on a tangent?
        Start your statement with    I   m  going on a tangent,    to explicitly
state
        that you are going on a tangent and intend to return to the main topic
        of the discussion after  y o u  r e  done.

NATURE OF THE MEETING
- - - - - - -
When does a meeting start?
```

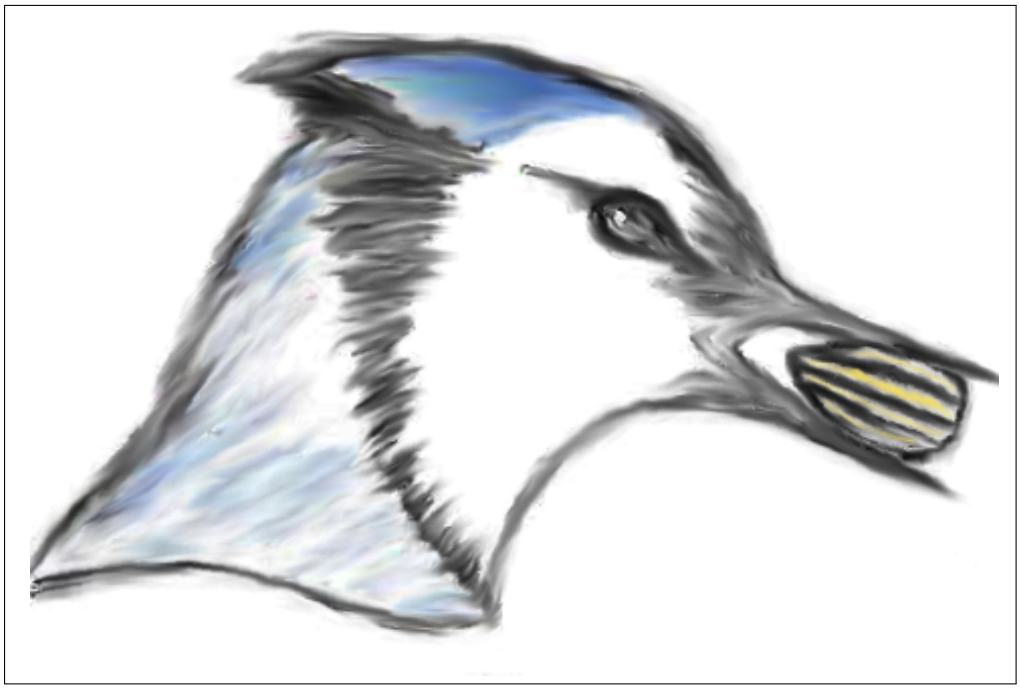
```
        A meeting starts when all members are in attendance,
        or 10 minutes after the original meeting time,
        whichever comes first.

FACILITATOR
- - - - - - -
Who facilitates meetings?
        ???
What should a facilitator do?
        A facilitator should guide discussions, keep the group on topic,
        and most importantly, ensure the group s success
        without having a stake in a specific decision.

NOTES
- - - - - - -
Who is taking notes?
        Currently, Josh is taking notes.
What if I want to see them live?
        The notetaker will screencast the notes so everyone can see them live.
Where can I find notes if I missed a meeting?
        Check this Subversion folder:
        /Documentation/Meeting Notes

AGENDA
- - - - - - -
What is the agenda s purpose?
       The agenda lists all items to cover during
       the meeting.
When should an agenda be published?
       Before the meeting.
What if I want to talk about something not on the agenda?
       If the facilitator allows it, you may talk about it.
       But try not to.

DECISIONS
- - - - - - -
How does a final decision work?
       Final decisions must be declared, and all team members
       must be consulted in a round robin fashion before
       finalizing the decision.
What if the team needs to revoke a final decision?
       In the same way it was declared: revoking the decision
       requires an appeal to revoke, followed by a
       consultation of all team members, round robin style.

ENDING A MEETING
- - - - - - -
When is this meeting gonna be OVER?
        When all the items on the agenda are finished,
        or the team finds a stopping point.

3. ONLINE MEETINGS
******************
Do I have to turn on my webcam?
      No. But we like seeing you.
When should I mute?
       If your mic feed is dirty, you should mute it
       when you re not speaking. Someone will complain about it.
```

```
What if somebody is breaking up?
       Tell them so they know, and they can repeat what they said.

4. SUBVERSION
*****************************
Who s maintaining the SVN directory right now?
         Currently, Thomas is managing it.
Which folders can I access?
      You can read any file from any folder.
      Subgroups have complete jurisdiction of the contents in their folder.
What is the sandbox folder for?
          It s 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

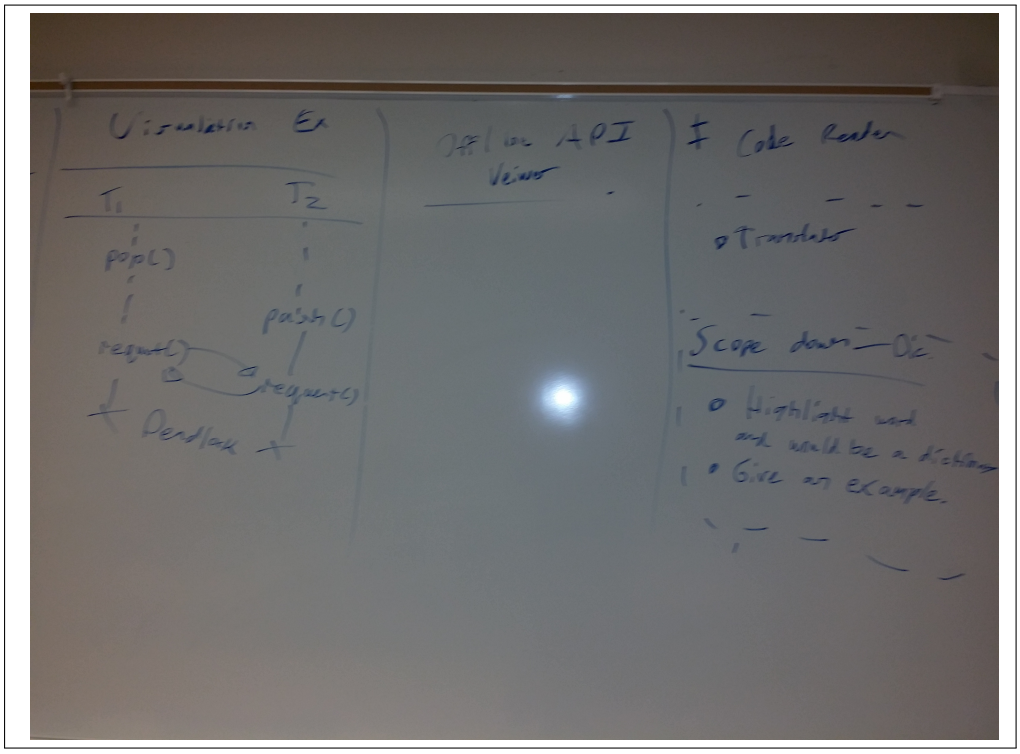
```
  BlueJ Git Extension
==========================================================================

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.
```

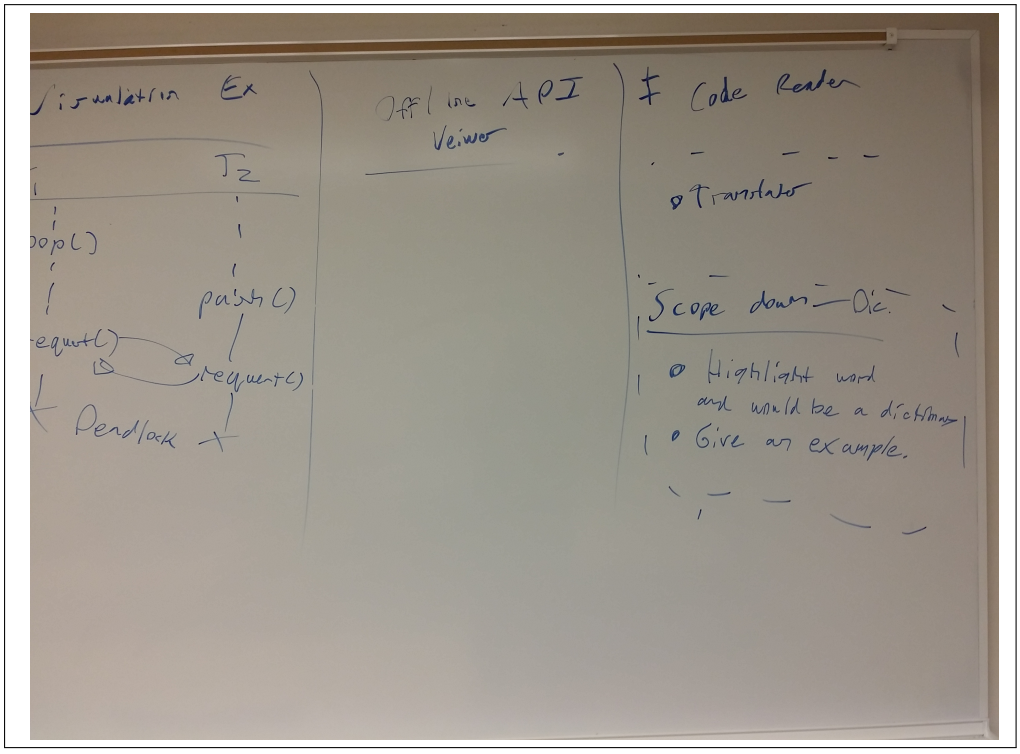
```
        Exceptions      --- Git Extension Exceptions.
         Helpers        --- Simplifies talking to BlueJ and JGit.

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

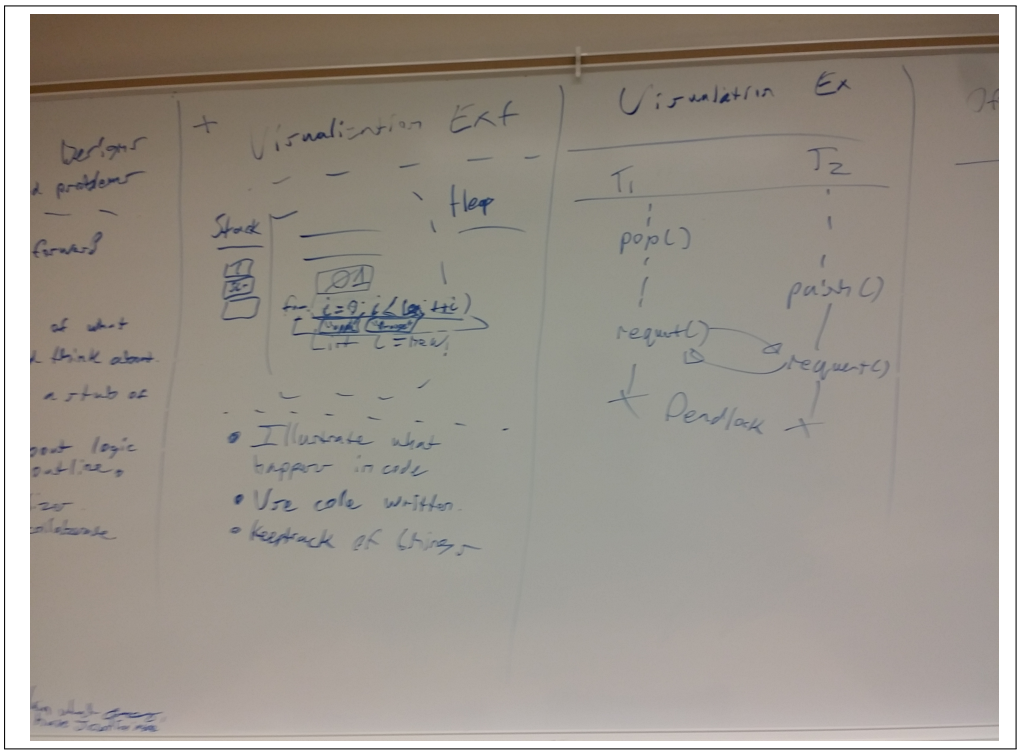
END File: /Projects/Git/Git-Extension/README.TXT

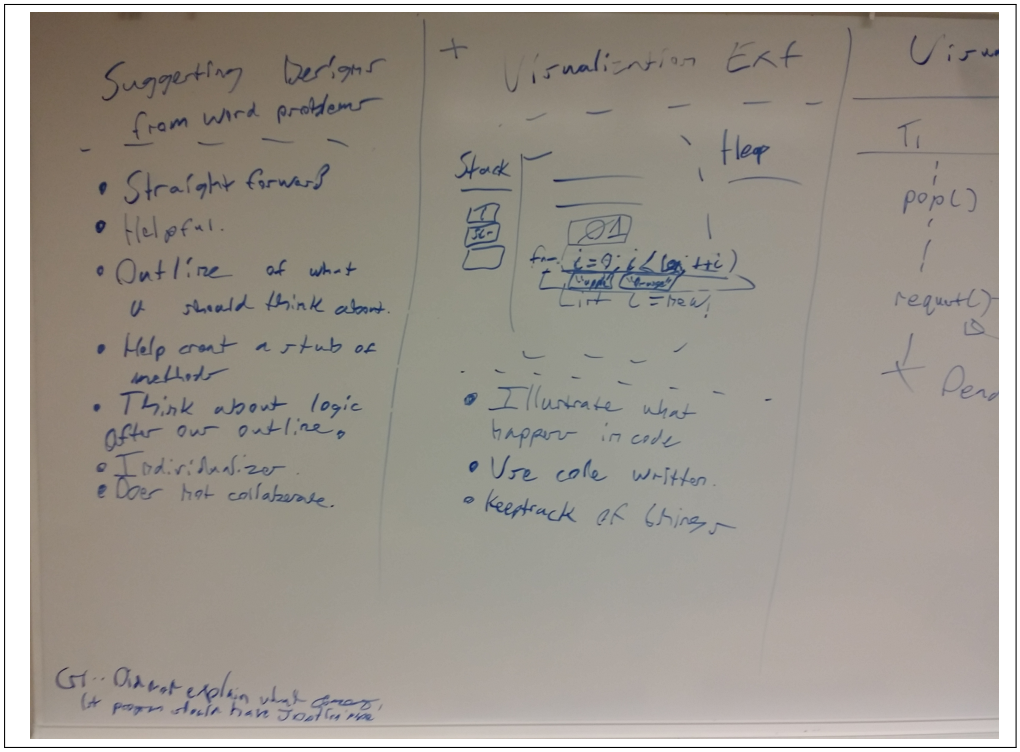
Surveys

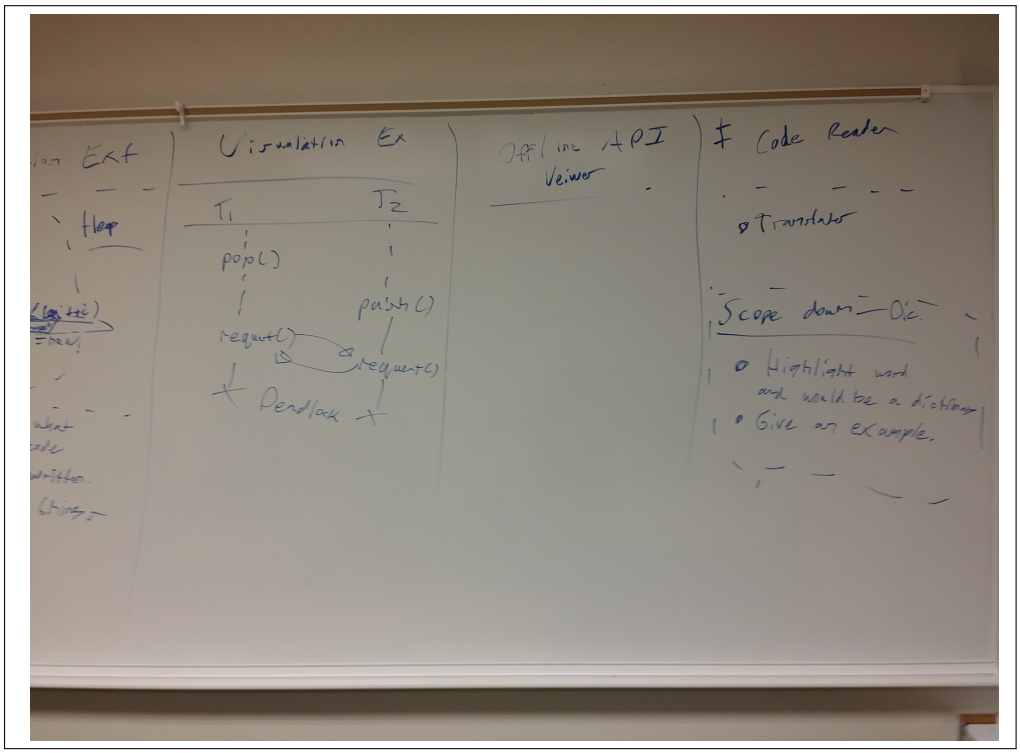
Description This survey was used to inquire students about their programming experience, their biggest challenge, what helped them overcome their biggest challenge, and additional comments. The survey also includes thoughts from 2 professors about issues students usually face, what made it easier for 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.

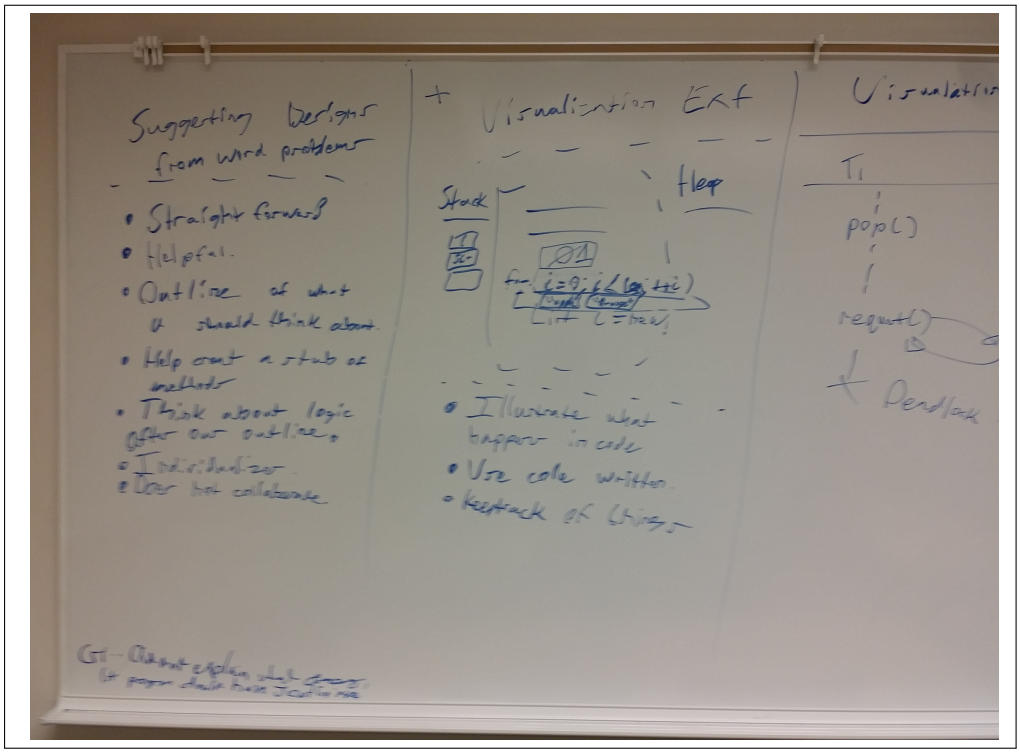


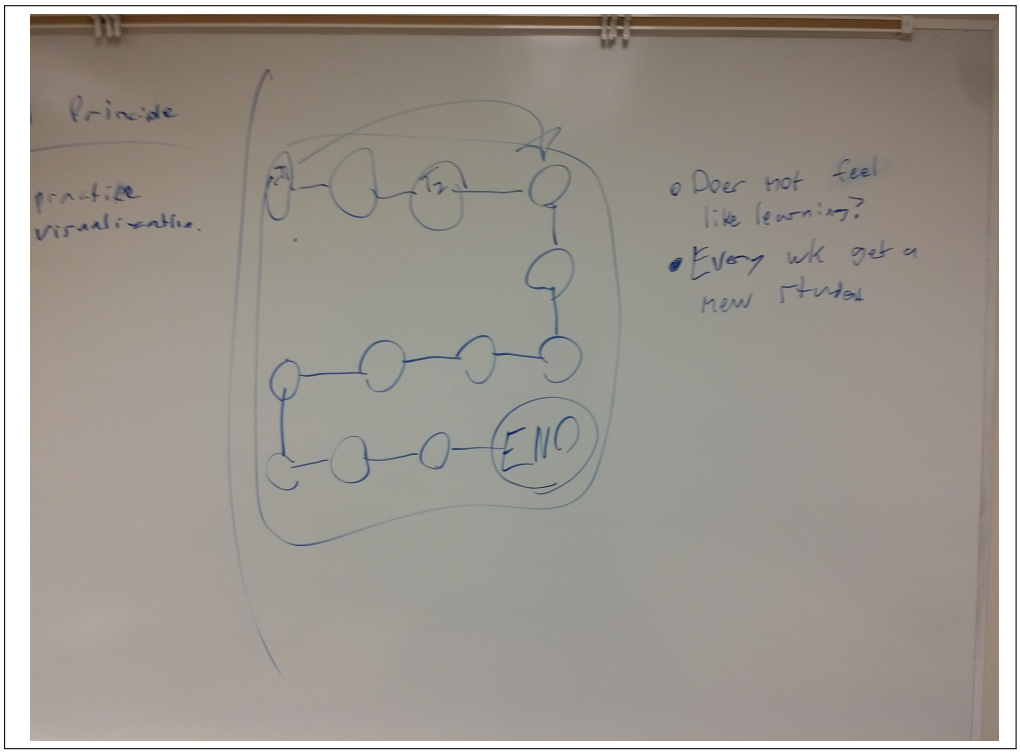


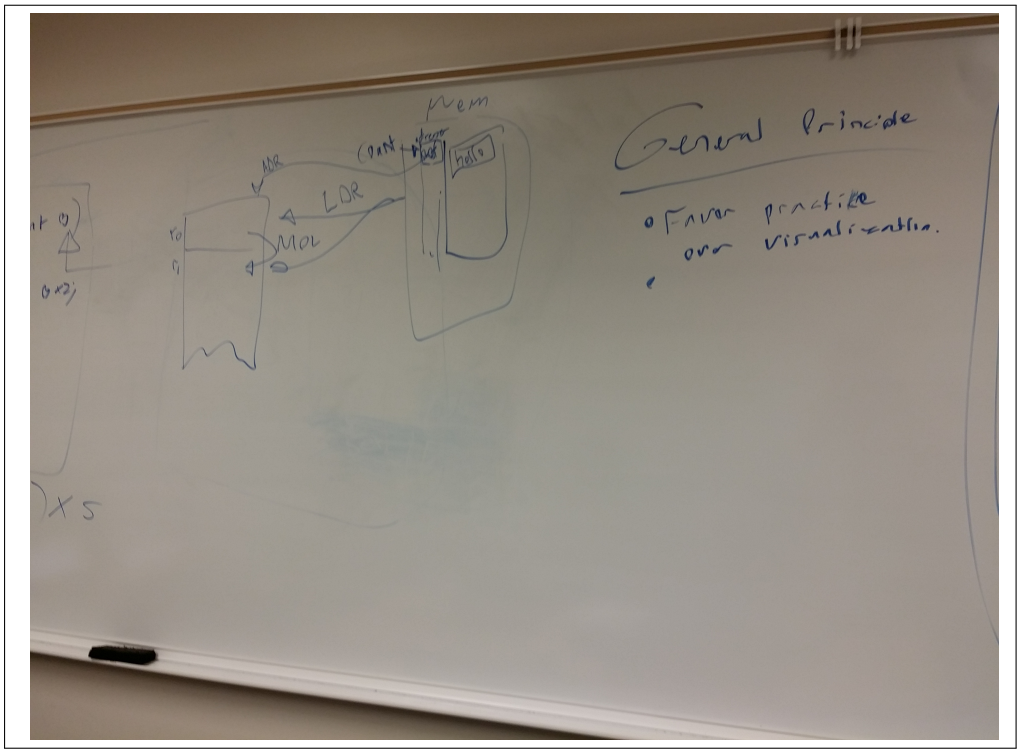

Visualization Ext
Stack
Heap
T1
T2
pop()
push()
request()
request()
Deadlock
Illustrate what happens in code
Use code written.
Keep track of things.

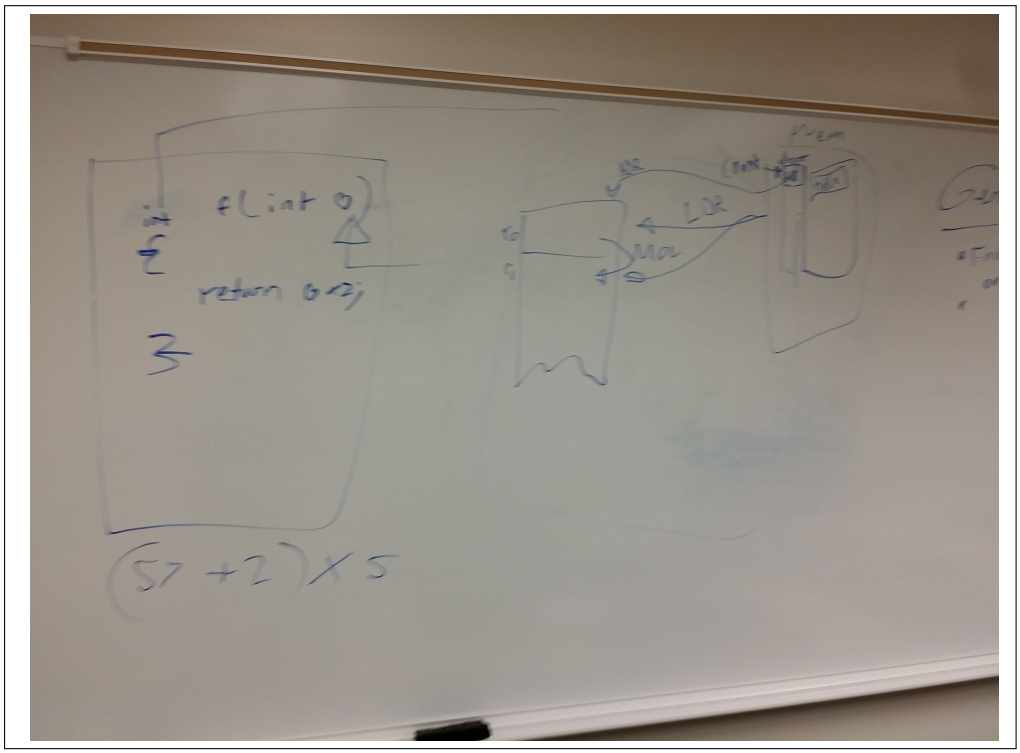

Suggesting Designs from word problems
Straight forward
Helpful.
Outline of what U should think about.
Help create a stub of methods
Think about logic after our outline.
Individualizer.
Does not collaborate.
Visualization Ext
Stack
Heap
T1
pop()
request()
Illustrate what happens in code
Use code written.
Keep track of things.


Visualization Ex
Offline API Viewer
Code Reader
Translator
Scope down — Oic
Highlight word and would be a dictionary
Give an example.
T1
T2
pop()
push()
request()
request()
Deadlock


Suggesting designs from word problems
Straight forward
Helpful.
Outline of what u should think about.
Help create a stub or methods
Think about logic after our outline.
Individualize
Draw that collaborate
Visualization Ext
Stack
Heap
Illustrate what happens in code
Use code written.
Keep track of Strings
Visualiz
T1
pop()
request()
Deadlock


Principle
practice
visualization.
END
Does not feel like learning?
Every wk get a new student


Mem
LDR
MOV
General Principle
Favor practice over visualization.


f(int a)
return a*5;
(57+2)*5

Students:

Year in College	Major	Programming Experience	Biggest Issue	What helped or would have helped	Comments:
Senior	CS	4 years	Syntax	N/A (First survey didn't ask)	Lack of feedback from instructors contributed to this.
Senior	CS	3 years	Syntax	Working in a group	It was difficult to say something in a way the language would understand.
Junior	CS	3 years	Switching programming paradigms	Learning why something works the way it does and how it works that way. (Concepts)	It was difficult because we are taught one way then switched to another way.
Sophomore	Other	1 semester	Syntax	A tutor	What would be cool would be a way to use voice to create programs.
Sophomore	CS	2 years	Lack of info and missing info	Visuals and examples of algorithms and structures	The Textbook for data structures aren't good. Lot of the info for the structures is missing.
Junior	CS	1.5	lack of standards in IDE.	Rebuilding existing programs.	N/A
Sophomore	CS	1 year	Recursion and concepts of data structures	Hands on experience.	More exposure to different languages would be great.
Junior	Former CS	2 years	Math and Syntax	1 on 1 help	Left program because of Math
Junior	Other	1 year	Java Syntax	visiting a tutor everyday.	I want a good tutor program that gives you practice on syntax.
Junior	Other	1 semester	Syntax, functions, and	A tutor	I want a dictionary for all the code.

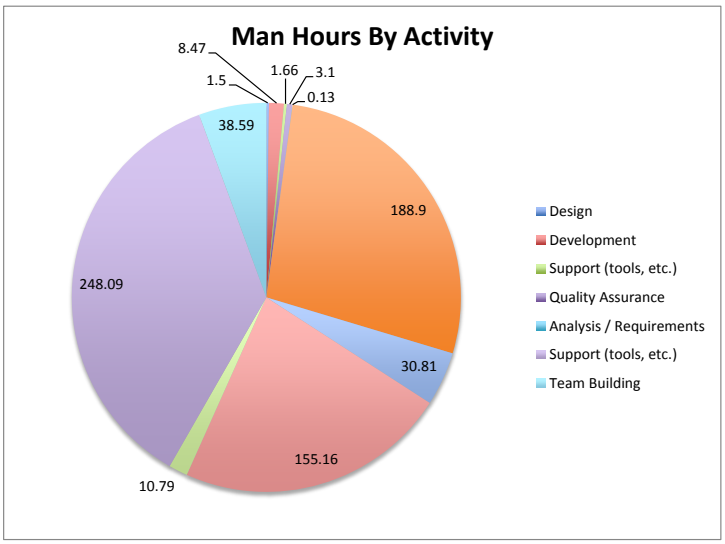
Metrics

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.

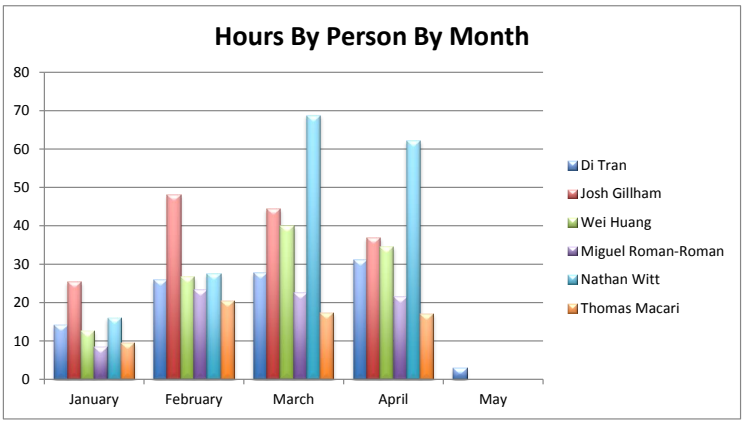
Timesheet

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

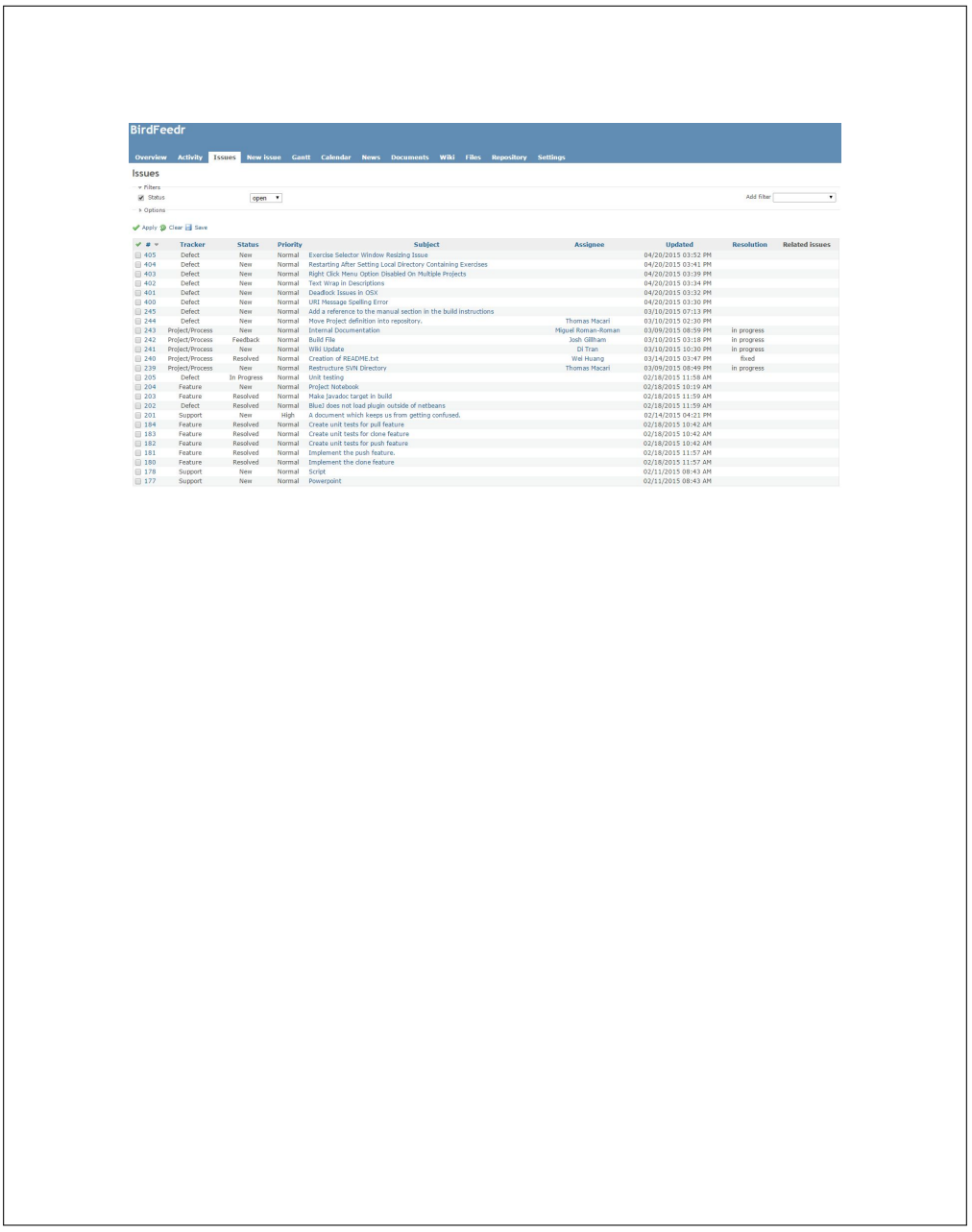


Name	January	February	March	April	May	Total time
Di Tran	14.25	25.91	27.7	31.15	3	102.01
Josh Gillham	25.38	48.14	44.36	36.93		154.8
Wei Huang	12.55	26.75	40	34.5		113.8
Miguel Roma	8.45	23.4	22.5	21.5		75.85
Nathan Witt	16	27.5	68.75	62		174.25
Thomas Maca	9.55	20.45	17.4	17.1		64.5
Total time	86.17	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
caution.
------------------------------------------------------------------------
r295 | jgillham | 2015-05-01 09:51:11 -0600 (Fri, 01 May 2015) | 1 line

Added script to clean up junk files.
------------------------------------------------------------------------
r294 | jgillham | 2015-05-01 09:50:54 -0600 (Fri, 01 May 2015) | 1 line

Added Google Drive file change report.
------------------------------------------------------------------------
r293 | jgillham | 2015-05-01 09:49:34 -0600 (Fri, 01 May 2015) | 1 line

Added Sandbox files from Google Drive.
------------------------------------------------------------------------
r292 | jgillham | 2015-05-01 09:48:30 -0600 (Fri, 01 May 2015) | 1 line

Cleaned up *.class *.ctxt in my sandbox.
------------------------------------------------------------------------
r291 | jgillham | 2015-05-01 09:46:09 -0600 (Fri, 01 May 2015) | 1 line

Added Team Documentation from Google Drive.
------------------------------------------------------------------------
r290 | jgillham | 2015-05-01 09:44:46 -0600 (Fri, 01 May 2015) | 1 line

Added project documentation from Google Drive.
------------------------------------------------------------------------
r289 | jgillham | 2015-05-01 09:35:49 -0600 (Fri, 01 May 2015) | 1 line

Uploaded Presentation 2.
```

```
------------------------------------------------------------------------
r288 | jgillham | 2015-05-01 09:29:13 -0600 (Fri, 01 May 2015) | 1 line

Added PDFs for Presentation 1.
------------------------------------------------------------------------
r287 | jgillham | 2015-05-01 09:27:27 -0600 (Fri, 01 May 2015) | 1 line

Uploaded Presentation 3.
------------------------------------------------------------------------
r286 | jgillham | 2015-05-01 09:19:20 -0600 (Fri, 01 May 2015) | 1 line

Moved folders into BlueJ TA Documentation.
------------------------------------------------------------------------
r285 | jgillham | 2015-05-01 09:10:48 -0600 (Fri, 01 May 2015) | 1 line

Deleted files in my own folder.
------------------------------------------------------------------------
r284 | nwitt | 2015-04-30 16:42:27 -0600 (Thu, 30 Apr 2015) | 1 line

Edited the examples in SortExercise.xml to be in brackets.
------------------------------------------------------------------------
r283 | jgillham | 2015-04-30 15:53:34 -0600 (Thu, 30 Apr 2015) | 1 line

Added screenshots showing GUI.
------------------------------------------------------------------------
r282 | nwitt | 2015-04-29 15:38:39 -0600 (Wed, 29 Apr 2015) | 1 line

Added some diagrams depicting execution of a test in BlueJ TA.
------------------------------------------------------------------------
r281 | dtran | 2015-04-29 00:05:34 -0600 (Wed, 29 Apr 2015) | 1 line

Added meeting notes for 4/27
------------------------------------------------------------------------
r280 | whuang | 2015-04-29 00:05:15 -0600 (Wed, 29 Apr 2015) | 1 line

Edited the Readme.txt file to better represent our SVN and a note about using
the ant build file instead of manually building the project.
------------------------------------------------------------------------
r279 | mroman | 2015-04-28 17:27:32 -0600 (Tue, 28 Apr 2015) | 1 line

created JavaDoc html documents. modified Javadoc for some classes
------------------------------------------------------------------------
r278 | jgillham | 2015-04-27 16:00:43 -0600 (Mon, 27 Apr 2015) | 1 line

Created a report for the lines of code of each project.
------------------------------------------------------------------------
r277 | dtran | 2015-04-25 16:31:29 -0600 (Sat, 25 Apr 2015) | 1 line

Added notes for 4/22
------------------------------------------------------------------------
r276 | dtran | 2015-04-25 16:30:16 -0600 (Sat, 25 Apr 2015) | 1 line

Added today's meeting notes.
------------------------------------------------------------------------
r275 | jgillham | 2015-04-23 12:58:27 -0600 (Thu, 23 Apr 2015) | 1 line

Fixed the build issue with Macs.
------------------------------------------------------------------------
```

```
r274 | jgillham | 2015-04-23 09:39:42 -0600 (Thu, 23 Apr 2015) | 1 line

Uploaded note book entries that summarize progress.
------------------------------------------------------------------------
r273 | mroman | 2015-04-22 13:47:39 -0600 (Wed, 22 Apr 2015) | 1 line

made small changes to Javadoc and renamed GUI windows
------------------------------------------------------------------------
r272 | mroman | 2015-04-22 13:25:13 -0600 (Wed, 22 Apr 2015) | 1 line

added Javadoc to most classes
------------------------------------------------------------------------
r271 | nwitt | 2015-04-22 13:05:45 -0600 (Wed, 22 Apr 2015) | 1 line

Touched up some of the documentation in the JRunner and remove thread classes.
------------------------------------------------------------------------
r270 | nwitt | 2015-04-22 02:52:51 -0600 (Wed, 22 Apr 2015) | 1 line

Fixed some documentation typos.
------------------------------------------------------------------------
r269 | nwitt | 2015-04-22 02:36:47 -0600 (Wed, 22 Apr 2015) | 3 lines

Added some documentation to the JRunner, Exercise, and FileUtil classes.
Also refactored the Exercises execute method and added the RemoveThread as part
of the refactoring.

------------------------------------------------------------------------
r268 | jgillham | 2015-04-21 18:55:03 -0600 (Tue, 21 Apr 2015) | 1 line

Reverted to older build code to ensure Macs can build.
------------------------------------------------------------------------
r267 | nwitt | 2015-04-21 15:50:36 -0600 (Tue, 21 Apr 2015) | 2 lines

Fixed the sort exercise bug.
Now checks ascending order as intended and the sample answer has been updated
to fit this change as well.
------------------------------------------------------------------------
r266 | jgillham | 2015-04-21 15:34:57 -0600 (Tue, 21 Apr 2015) | 1 line

Major update to the build.xml. Linux support added.
------------------------------------------------------------------------
r265 | jgillham | 2015-04-20 16:04:30 -0600 (Mon, 20 Apr 2015) | 1 line

Working on Java files to be outside of the xml.
------------------------------------------------------------------------
r264 | dtran | 2015-04-20 16:02:32 -0600 (Mon, 20 Apr 2015) | 1 line

Added meeting notes for 4/20
------------------------------------------------------------------------
r263 | jgillham | 2015-04-20 15:36:04 -0600 (Mon, 20 Apr 2015) | 1 line

Build uses special compile for the JRunner.
------------------------------------------------------------------------
r262 | jgillham | 2015-04-20 15:01:16 -0600 (Mon, 20 Apr 2015) | 1 line

The compiled class needs to moved back into the secret folder.
------------------------------------------------------------------------
r261 | jgillham | 2015-04-20 14:57:05 -0600 (Mon, 20 Apr 2015) | 1 line
```

```
Moved JRunner into the secret folder so that new developers are not confused.
------------------------------------------------------------------------
r260 | jgillham | 2015-04-20 14:39:00 -0600 (Mon, 20 Apr 2015) | 1 line

Renamed project folder and moved in new refactor.
------------------------------------------------------------------------
r259 | jgillham | 2015-04-20 14:03:09 -0600 (Mon, 20 Apr 2015) | 1 line

Updated build for new refactor.
------------------------------------------------------------------------
r258 | nwitt | 2015-04-20 04:50:57 -0600 (Mon, 20 Apr 2015) | 1 line


------------------------------------------------------------------------
r257 | nwitt | 2015-04-20 04:43:50 -0600 (Mon, 20 Apr 2015) | 14 lines

Added a re-factored version of the BlueJ TA project to my sandbox.

Major Changes(from vers A):
- Removed the runner package and extended the Exercise.java's execute method.
- Added a JRunner class to be deployed during execution.
- Test should now be able to launch and execute on windows and macs*.
(tested on one mac to be tested on more macs later).
- New feature allows test to be stopped by reseting BlueJ's jvm.
- New feature has less code to maintain.
- Currently no build or bat files just the BlueJ project, to hopefully use
joshes later.

To install compile the .java files and move the JRunner.class file from the
default package to the Extension/BackEnd/runner folder, then create a jar as
normal.
(this is beacue the JRunner cant have a package header in order to work)

------------------------------------------------------------------------
r256 | nwitt | 2015-04-19 13:44:10 -0600 (Sun, 19 Apr 2015) | 1 line

Switched the class.forname loading method in the TestRunner.java class to a url
class loader and edited the JUnitRunner.java to accommodate this change to
hopefully fix the mac execution issues.
------------------------------------------------------------------------
r255 | jgillham | 2015-04-18 19:34:25 -0600 (Sat, 18 Apr 2015) | 1 line

Copied build files to the main folder.
------------------------------------------------------------------------
r254 | jgillham | 2015-04-18 19:29:34 -0600 (Sat, 18 Apr 2015) | 1 line

Cleaned up and documented build.xml.
------------------------------------------------------------------------
r253 | nwitt | 2015-04-18 19:18:12 -0600 (Sat, 18 Apr 2015) | 1 line

Edited JUnitRunner.java to hopefully fix execute issues.
------------------------------------------------------------------------
r252 | nwitt | 2015-04-18 19:04:58 -0600 (Sat, 18 Apr 2015) | 7 lines

Moved BlueJ_TA(RefactorA) to the main project folder.
Fixes:
- SortExercise.xml now test ascending order.
```

```
- Exercise now launch on mac and windows.

Bugs:
- The execute test feature doesn't work on macs because the junit test file
cant be seen by the JUnitRunners process. Currently looking into the -cp
variable of the process builder in JUnitRunner.java.
------------------------------------------------------------------------
r251 | jgillham | 2015-04-18 17:26:41 -0600 (Sat, 18 Apr 2015) | 1 line

build.xml launches a BlueJ on Mac better and the error on windows should be
gone.
------------------------------------------------------------------------
r250 | jgillham | 2015-04-18 16:40:16 -0600 (Sat, 18 Apr 2015) | 1 line

build.xml should work on Mac.
------------------------------------------------------------------------
r249 | jgillham | 2015-04-18 16:20:13 -0600 (Sat, 18 Apr 2015) | 1 line

Fixed build.xml to work on MacOSX.
------------------------------------------------------------------------
r248 | jgillham | 2015-04-18 16:12:44 -0600 (Sat, 18 Apr 2015) | 1 line

Fixed the build to work in more than one platform.
------------------------------------------------------------------------
r247 | dtran | 2015-04-18 15:58:11 -0600 (Sat, 18 Apr 2015) | 1 line

Added meeting notes for 4/18
------------------------------------------------------------------------
r246 | jgillham | 2015-04-18 15:56:05 -0600 (Sat, 18 Apr 2015) | 1 line

Created a build for Nate's refactor.
------------------------------------------------------------------------
r245 | nwitt | 2015-04-16 19:56:15 -0600 (Thu, 16 Apr 2015) | 1 line

Added A re-factored version of the BlueJ TA program. and added Java doc to the
JUnitRunner classes.
------------------------------------------------------------------------
r244 | jgillham | 2015-04-16 17:30:46 -0600 (Thu, 16 Apr 2015) | 1 line

Upgraded the Windows batch files.
------------------------------------------------------------------------
r243 | jgillham | 2015-04-16 10:50:17 -0600 (Thu, 16 Apr 2015) | 1 line

The extension will load exercises internal to the JAR.
------------------------------------------------------------------------
r242 | nwitt | 2015-04-15 21:51:51 -0600 (Wed, 15 Apr 2015) | 1 line

Removed class and ctxt files from the runner and made the result output more
readable.
------------------------------------------------------------------------
r241 | nwitt | 2015-04-15 21:38:52 -0600 (Wed, 15 Apr 2015) | 1 line

removed dist for now
------------------------------------------------------------------------
r240 | nwitt | 2015-04-15 21:34:32 -0600 (Wed, 15 Apr 2015) | 1 line

Added a dist folder for the extension jar
------------------------------------------------------------------------
```

```
r239 | nwitt | 2015-04-15 21:31:30 -0600 (Wed, 15 Apr 2015) | 4 lines

A) Made it so examples are displayed in the guis.

B) Refactored out coding actions and replaced it with just AbstractAction.

------------------------------------------------------------------------
r238 | nwitt | 2015-04-15 19:06:44 -0600 (Wed, 15 Apr 2015) | 1 line


------------------------------------------------------------------------
r237 | nwitt | 2015-04-15 19:06:30 -0600 (Wed, 15 Apr 2015) | 1 line

Typo in last message should be: "Exercises now launch when selected from the
exercise list."
------------------------------------------------------------------------
r236 | nwitt | 2015-04-15 19:00:53 -0600 (Wed, 15 Apr 2015) | 1 line

Exercises now not launch when selected from the exercise list.
------------------------------------------------------------------------
r235 | jgillham | 2015-04-15 17:51:23 -0600 (Wed, 15 Apr 2015) | 1 line

Working on refactoring.
------------------------------------------------------------------------
r234 | dtran | 2015-04-15 17:30:51 -0600 (Wed, 15 Apr 2015) | 1 line

Added team meeting notes for 4/15
------------------------------------------------------------------------
r233 | nwitt | 2015-04-15 15:41:36 -0600 (Wed, 15 Apr 2015) | 1 line

Fixed Preferences.java
------------------------------------------------------------------------
r232 | nwitt | 2015-04-15 15:26:00 -0600 (Wed, 15 Apr 2015) | 1 line

Updated project, with Miguel and Thomas xml parser semei integrated in.
------------------------------------------------------------------------
r231 | mroman | 2015-04-15 15:06:37 -0600 (Wed, 15 Apr 2015) | 1 line

fixed project. compiles now. possibly broke gui.
------------------------------------------------------------------------
r230 | nwitt | 2015-04-14 16:31:47 -0600 (Tue, 14 Apr 2015) | 1 line

Fixed spelling error
------------------------------------------------------------------------
r229 | nwitt | 2015-04-14 16:21:39 -0600 (Tue, 14 Apr 2015) | 1 line

Re added SortExercise.xml with fixes.
------------------------------------------------------------------------
r228 | nwitt | 2015-04-14 15:56:47 -0600 (Tue, 14 Apr 2015) | 1 line

Removed SortExercise.xml to be fixed
------------------------------------------------------------------------
r227 | nwitt | 2015-04-14 15:41:55 -0600 (Tue, 14 Apr 2015) | 1 line

Added two simple exercises to the exercise folder, complete with the code the
program needs to make and run them.
------------------------------------------------------------------------
r226 | jgillham | 2015-04-13 23:11:38 -0600 (Mon, 13 Apr 2015) | 8 lines
```

```
The build works on both windows and MAC with mixed revisions (see details).

Details:
The build is currently broken unless Exercises.java
and FXMLDescriptionDocumentController.java are on revision 218.



------------------------------------------------------------------------
r225 | dtran | 2015-04-13 21:23:47 -0600 (Mon, 13 Apr 2015) | 1 line

Added meeting notes for 4/13.
------------------------------------------------------------------------
r224 | jgillham | 2015-04-13 17:54:54 -0600 (Mon, 13 Apr 2015) | 1 line

More.
------------------------------------------------------------------------
r223 | jgillham | 2015-04-13 17:54:37 -0600 (Mon, 13 Apr 2015) | 1 line

More.
------------------------------------------------------------------------
r222 | jgillham | 2015-04-13 17:53:47 -0600 (Mon, 13 Apr 2015) | 1 line

Latest changes.
------------------------------------------------------------------------
r221 | jgillham | 2015-04-13 16:42:27 -0600 (Mon, 13 Apr 2015) | 1 line

Reverted working copy.
------------------------------------------------------------------------
r220 | jgillham | 2015-04-13 16:05:00 -0600 (Mon, 13 Apr 2015) | 1 line

Committed work in progress.
------------------------------------------------------------------------
r219 | tmacari | 2015-04-13 15:32:58 -0600 (Mon, 13 Apr 2015) | 1 line

Refactored XML Reader
------------------------------------------------------------------------
r218 | jgillham | 2015-04-13 14:49:29 -0600 (Mon, 13 Apr 2015) | 1 line

Updated Mac build.
------------------------------------------------------------------------
r217 | dtran | 2015-04-11 16:40:28 -0600 (Sat, 11 Apr 2015) | 1 line

Added meeting notes from name meeting
------------------------------------------------------------------------
r216 | dtran | 2015-04-11 15:12:32 -0600 (Sat, 11 Apr 2015) | 1 line

Changed date and made small modifications to end of notes for 4/8.
------------------------------------------------------------------------
r215 | dtran | 2015-04-11 15:11:59 -0600 (Sat, 11 Apr 2015) | 1 line

Added team meeting notes for 4/11.
------------------------------------------------------------------------
r214 | jgillham | 2015-04-11 14:42:23 -0600 (Sat, 11 Apr 2015) | 1 line

Added clean script.
------------------------------------------------------------------------
```

```
r213 | jgillham | 2015-04-10 00:24:42 -0600 (Fri, 10 Apr 2015) | 1 line

Updated the install script for Windows.
------------------------------------------------------------------------
r212 | nwitt | 2015-04-09 22:48:50 -0600 (Thu, 09 Apr 2015) | 5 lines

Integrated what happens after the load button is pressed into the main project.
with a hard coded exercise to be replaced after the xml parser is updated.

Also Refractored the JUnit runner classes into their own package.
Note) the windows .bat build doesn't wont work with the nested folder but i
think this can be fixed relatively easily, the project can still be built using
the export jar methods in bluej in the meantime)

------------------------------------------------------------------------
r211 | nwitt | 2015-04-08 22:10:45 -0600 (Wed, 08 Apr 2015) | 1 line

Removed obsolete fragments around the source code.
------------------------------------------------------------------------
r210 | dtran | 2015-04-08 21:16:49 -0600 (Wed, 08 Apr 2015) | 1 line

Added meeting minutes for 4/8 class time.
------------------------------------------------------------------------
r209 | jgillham | 2015-04-08 15:52:08 -0600 (Wed, 08 Apr 2015) | 1 line

Windows install script now compiles.
------------------------------------------------------------------------
r208 | mroman | 2015-04-08 15:34:07 -0600 (Wed, 08 Apr 2015) | 1 line

Added our extension to the Preferences tab in BlueJ. Users can now set the
location of the exercises using this BLuej->Preferences->Extensions. The path
is saved into StateManager.
------------------------------------------------------------------------
r207 | jgillham | 2015-04-08 14:56:35 -0600 (Wed, 08 Apr 2015) | 1 line

Script installs extension on Windows.
------------------------------------------------------------------------
r206 | nwitt | 2015-04-08 13:05:35 -0600 (Wed, 08 Apr 2015) | 1 line

Edited small feature in prototype exercise file, to hide the root pane when the
project opens so the user cant see the test class compiling.
------------------------------------------------------------------------
r205 | nwitt | 2015-04-08 02:31:48 -0600 (Wed, 08 Apr 2015) | 1 line

Added a prototype extension that demonstrates what could happen after the load
button is pressed, in this case when the launch menu item is pressed. (Will go
over it in class if asked.)
------------------------------------------------------------------------
r204 | dtran | 2015-04-06 22:44:20 -0600 (Mon, 06 Apr 2015) | 1 line

Uploaded notes for Monday, April 6th.
------------------------------------------------------------------------
r203 | jgillham | 2015-04-06 16:52:51 -0600 (Mon, 06 Apr 2015) | 1 line

Created a script to install and build the program on mac.
------------------------------------------------------------------------
r202 | dtran | 2015-04-04 15:30:29 -0600 (Sat, 04 Apr 2015) | 1 line
```

```
Added team meeting notes for april 4th, which includes Monday's agenda and
rough status update.
------------------------------------------------------------------------
r201 | mroman | 2015-04-04 11:03:40 -0600 (Sat, 04 Apr 2015) | 1 line

added my design ideas and created a small list of known issues/bugs
------------------------------------------------------------------------
r200 | jgillham | 2015-04-03 15:11:02 -0600 (Fri, 03 Apr 2015) | 1 line

Adding extension xml's.
------------------------------------------------------------------------
r199 | jgillham | 2015-04-03 14:45:48 -0600 (Fri, 03 Apr 2015) | 1 line

Created a Windows batch file that installs.
------------------------------------------------------------------------
r198 | nwitt | 2015-04-01 23:35:55 -0600 (Wed, 01 Apr 2015) | 1 line

Added some diagrams and examples of how files can be stored by the extension in
the local user lib, as well as what can happen after the load button is pressed.
------------------------------------------------------------------------
r197 | nwitt | 2015-04-01 20:29:32 -0600 (Wed, 01 Apr 2015) | 3 lines

Edited Temp_Instructions.txt to accommodate creating the test class setup (this
should be done programmatically by the extension at some point)

As well as edited the RunnerThread.java for more descriptive error messages.
------------------------------------------------------------------------
r196 | nwitt | 2015-04-01 19:03:28 -0600 (Wed, 01 Apr 2015) | 3 lines

Integrated the JUnit runner into the main project (note: hasn't been fully
debugged yet, might have an unseen bugs)

Read the Temp_Instructions.txt for more info on running a "test" test.
------------------------------------------------------------------------
r195 | mroman | 2015-04-01 16:21:46 -0600 (Wed, 01 Apr 2015) | 1 line

Integrated parser. Also implemented opening of new BlueJ project
------------------------------------------------------------------------
r194 | nwitt | 2015-03-30 23:39:54 -0600 (Mon, 30 Mar 2015) | 1 line

Added the final demo runner to sandbox
------------------------------------------------------------------------
r193 | nwitt | 2015-03-30 23:35:56 -0600 (Mon, 30 Mar 2015) | 1 line

Refined JunitRunner down to the main classes to add/replace the old JunitRunner
```

***** Lines 500-831 were ommitted in this abridged version. *****

```
------------------------------------------------------------------------
r119 | jgillham | 2015-03-11 09:30:40 -0600 (Wed, 11 Mar 2015) | 1 line

Ignoring class files.
------------------------------------------------------------------------
r118 | jgillham | 2015-03-11 09:29:59 -0600 (Wed, 11 Mar 2015) | 1 line

Removed generated by-products.
------------------------------------------------------------------------
r117 | jgillham | 2015-03-11 09:26:42 -0600 (Wed, 11 Mar 2015) | 1 line
```

```
Updated Git-Extension README.TXT.
------------------------------------------------------------------------
r116 | jgillham | 2015-03-11 09:08:12 -0600 (Wed, 11 Mar 2015) | 1 line

Added installation instructions for Git-Extension.
------------------------------------------------------------------------
r115 | jgillham | 2015-03-11 09:07:38 -0600 (Wed, 11 Mar 2015) | 1 line

Removed build.xml in favor of BlueJ's build in Git-Extension.
------------------------------------------------------------------------
r114 | jgillham | 2015-03-11 08:55:12 -0600 (Wed, 11 Mar 2015) | 1 line

Removed sample project.
------------------------------------------------------------------------
r113 | jgillham | 2015-03-11 08:53:45 -0600 (Wed, 11 Mar 2015) | 1 line

Removed build.xml in favor of using BlueJ to build.
------------------------------------------------------------------------
r112 | whuang | 2015-03-10 23:57:08 -0600 (Tue, 10 Mar 2015) | 1 line

The Readme.txt that explains our current project, how to build a project
manually, and the structure of the repository.
------------------------------------------------------------------------
r111 | jgillham | 2015-03-10 14:18:23 -0600 (Tue, 10 Mar 2015) | 1 line

Added a file with directions on how to install the extension.
------------------------------------------------------------------------
r110 | jgillham | 2015-03-10 13:48:06 -0600 (Tue, 10 Mar 2015) | 1 line

Using BlueJ to build the project.
------------------------------------------------------------------------
r109 | jgillham | 2015-03-09 15:56:58 -0600 (Mon, 09 Mar 2015) | 1 line

New meeting notes.
------------------------------------------------------------------------
r108 | jgillham | 2015-03-09 15:31:04 -0600 (Mon, 09 Mar 2015) | 1 line

Added a build file.
------------------------------------------------------------------------
r107 | nwitt | 2015-03-09 15:11:30 -0600 (Mon, 09 Mar 2015) | 1 line

Added GUI test
------------------------------------------------------------------------
r106 | mroman | 2015-03-09 14:49:53 -0600 (Mon, 09 Mar 2015) | 1 line

Miguel's implementation of JavaFX for the Coding Practice Project
------------------------------------------------------------------------
r105 | whuang | 2015-03-09 14:33:43 -0600 (Mon, 09 Mar 2015) | 1 line


------------------------------------------------------------------------
r104 | whuang | 2015-03-09 14:32:07 -0600 (Mon, 09 Mar 2015) | 1 line


------------------------------------------------------------------------
r103 | whuang | 2015-03-09 14:18:27 -0600 (Mon, 09 Mar 2015) | 1 line
```

```
------------------------------------------------------------------------
r102 | jgillham | 2015-03-07 16:31:40 -0700 (Sat, 07 Mar 2015) | 1 line

Uploaded today's meeting notes.
------------------------------------------------------------------------
r101 | whuang | 2015-03-06 16:48:58 -0700 (Fri, 06 Mar 2015) | 8 lines

Bluej exercise extension eclipse project folder. Also contains a Bluej package
to open the files in Bluej.
Java files located in the src folder.
Files included:
CodingAction.java - default action.
CodingExercisesAction.java - Action that opens ExerciseGUI.
ExerciseGUI.java - The GUI that will be the exercises.
Main.java - the startup for the extension.
MenuBuilder.java - Adds items to the bluej menu.
------------------------------------------------------------------------
r100 | jgillham | 2015-03-05 13:20:46 -0700 (Thu, 05 Mar 2015) | 1 line

Initial class definitions for BlueJ CodingBat Project.
------------------------------------------------------------------------
r99 | jgillham | 2015-03-05 13:19:26 -0700 (Thu, 05 Mar 2015) | 1 line

Uploaded meeting notes.
------------------------------------------------------------------------
r98 | nwitt | 2015-03-05 00:12:54 -0700 (Thu, 05 Mar 2015) | 2 lines

Added a simple multi window example for proof of concept, and for fun.
Note: The other approach of just one window for the whole extension is still
just as valid.
------------------------------------------------------------------------
r97 | whuang | 2015-03-02 21:06:58 -0700 (Mon, 02 Mar 2015) | 2 lines

Adding Design sketches.pdf to Wei's sandbox

------------------------------------------------------------------------
r96 | jgillham | 2015-02-28 22:20:53 -0700 (Sat, 28 Feb 2015) | 1 line

Uploaded new meeting notes.
------------------------------------------------------------------------
r95 | jgillham | 2015-02-25 10:50:58 -0700 (Wed, 25 Feb 2015) | 1 line

Informal report on Thread visualization project.
------------------------------------------------------------------------
r94 | jgillham | 2015-02-25 09:38:46 -0700 (Wed, 25 Feb 2015) | 1 line

Java Interpreter Informal Report.
------------------------------------------------------------------------
r93 | jgillham | 2015-02-24 21:52:06 -0700 (Tue, 24 Feb 2015) | 1 line

Animated definition slide.
------------------------------------------------------------------------
r92 | nwitt | 2015-02-24 20:41:08 -0700 (Tue, 24 Feb 2015) | 1 line

Fixed "whats pair slide"
------------------------------------------------------------------------
r91 | mroman | 2015-02-24 19:37:24 -0700 (Tue, 24 Feb 2015) | 1 line
```

```
modified narrator script
------------------------------------------------------------------------
r90 | jgillham | 2015-02-24 16:58:10 -0700 (Tue, 24 Feb 2015) | 1 line

Added comments to scripts and in-document comments on presentation.
------------------------------------------------------------------------
r89 | jgillham | 2015-02-24 14:35:37 -0700 (Tue, 24 Feb 2015) | 1 line

Added QA Notes on Presentation.
------------------------------------------------------------------------
r88 | nwitt | 2015-02-23 23:04:42 -0700 (Mon, 23 Feb 2015) | 1 line

Edited presentation power point.
------------------------------------------------------------------------
r87 | nwitt | 2015-02-23 22:43:13 -0700 (Mon, 23 Feb 2015) | 1 line

Renamed presentation slides.
------------------------------------------------------------------------
r86 | nwitt | 2015-02-23 22:42:38 -0700 (Mon, 23 Feb 2015) | 1 line

Edited Wrap-up slide of the presentation slides.
------------------------------------------------------------------------
r85 | nwitt | 2015-02-23 18:18:55 -0700 (Mon, 23 Feb 2015) | 1 line

Edited and Updated presentation slides.
------------------------------------------------------------------------
r84 | jgillham | 2015-02-23 16:20:03 -0700 (Mon, 23 Feb 2015) | 1 line

Added new meeting notes.
------------------------------------------------------------------------
r83 | mroman | 2015-02-23 09:34:32 -0700 (Mon, 23 Feb 2015) | 1 line

Added intro and conclusion points
------------------------------------------------------------------------
r82 | dtran | 2015-02-23 00:03:03 -0700 (Mon, 23 Feb 2015) | 1 line

Added negative bullet points.
------------------------------------------------------------------------
r81 | nwitt | 2015-02-21 20:19:47 -0700 (Sat, 21 Feb 2015) | 1 line

Edited good script.
------------------------------------------------------------------------
r80 | nwitt | 2015-02-21 20:17:00 -0700 (Sat, 21 Feb 2015) | 1 line

Organized good teams script files.
------------------------------------------------------------------------
r79 | nwitt | 2015-02-21 20:12:27 -0700 (Sat, 21 Feb 2015) | 1 line

Added positive points to the presentation as well as their animations.
------------------------------------------------------------------------
r78 | nwitt | 2015-02-21 15:37:12 -0700 (Sat, 21 Feb 2015) | 1 line

Edited goodScript
------------------------------------------------------------------------
r77 | nwitt | 2015-02-21 12:19:04 -0700 (Sat, 21 Feb 2015) | 1 line

Added Good team script 2.
```

```
------------------------------------------------------------------------
r76 | tmacari | 2015-02-21 11:16:38 -0700 (Sat, 21 Feb 2015) | 2 lines

Flushed out the narrator script more.

------------------------------------------------------------------------
r75 | tmacari | 2015-02-21 10:14:01 -0700 (Sat, 21 Feb 2015) | 1 line

Ported over documents from Google Drive.
------------------------------------------------------------------------
r74 | tmacari | 2015-02-21 10:10:35 -0700 (Sat, 21 Feb 2015) | 1 line

Created a directory for presentation artifacts.
------------------------------------------------------------------------
r73 | jgillham | 2015-02-20 14:31:47 -0700 (Fri, 20 Feb 2015) | 1 line

Added new meeting notes.
------------------------------------------------------------------------
r72 | jgillham | 2015-02-14 16:34:15 -0700 (Sat, 14 Feb 2015) | 1 line

Added today's meeting notes.
------------------------------------------------------------------------
r71 | dtran | 2015-02-14 16:17:40 -0700 (Sat, 14 Feb 2015) | 1 line

Added final decision declaration and revocation policy.
------------------------------------------------------------------------
r70 | dtran | 2015-02-14 16:09:06 -0700 (Sat, 14 Feb 2015) | 1 line

Added tangent policy.
------------------------------------------------------------------------
r69 | jgillham | 2015-02-13 09:09:57 -0700 (Fri, 13 Feb 2015) | 1 line

Created a test to show code can be lost in a merge. Currently fails.
------------------------------------------------------------------------
r68 | jgillham | 2015-02-12 14:18:20 -0700 (Thu, 12 Feb 2015) | 1 line

Divided tests by how many repositories they use to reduce the lines of code.
------------------------------------------------------------------------
r67 | jgillham | 2015-02-12 13:56:02 -0700 (Thu, 12 Feb 2015) | 1 line

Created Git Pull test that focuses on local repositories.
------------------------------------------------------------------------
r66 | jgillham | 2015-02-12 13:04:11 -0700 (Thu, 12 Feb 2015) | 1 line

Improved the gitPush test by focusing on local repositories. Test currently
fails.
------------------------------------------------------------------------
r65 | jgillham | 2015-02-12 12:25:44 -0700 (Thu, 12 Feb 2015) | 1 line

Improved the gitClone test by focusing on local repositories.
------------------------------------------------------------------------
r64 | jgillham | 2015-02-11 23:23:10 -0700 (Wed, 11 Feb 2015) | 1 line

Created a Java documentation target.
------------------------------------------------------------------------
r63 | jgillham | 2015-02-11 15:47:36 -0700 (Wed, 11 Feb 2015) | 1 line

Fixed issue with exception handling in the actionPerformed method.
```

```
------------------------------------------------------------------------
r62 | jgillham | 2015-02-11 15:40:33 -0700 (Wed, 11 Feb 2015) | 1 line

Fixed GitHelper Pull section.
------------------------------------------------------------------------
r61 | jgillham | 2015-02-11 15:32:29 -0700 (Wed, 11 Feb 2015) | 1 line

Added a Pull Command menu item.
------------------------------------------------------------------------
r60 | jgillham | 2015-02-11 15:28:09 -0700 (Wed, 11 Feb 2015) | 1 line

Added Pull command action, settings action was removed.
------------------------------------------------------------------------
r59 | jgillham | 2015-02-11 15:14:24 -0700 (Wed, 11 Feb 2015) | 1 line

Added Pull command.
------------------------------------------------------------------------
r58 | jgillham | 2015-02-11 11:57:15 -0700 (Wed, 11 Feb 2015) | 1 line

Created a README.md file to help new developers get started.
------------------------------------------------------------------------
r57 | jgillham | 2015-02-11 11:05:03 -0700 (Wed, 11 Feb 2015) | 1 line

Documented the build.xml.
------------------------------------------------------------------------
r56 | jgillham | 2015-02-11 09:57:30 -0700 (Wed, 11 Feb 2015) | 1 line

Moved debug property out of the build to increased its visibility.
------------------------------------------------------------------------
r55 | jgillham | 2015-02-10 13:49:46 -0700 (Tue, 10 Feb 2015) | 1 line

The compilation source and target levels should be set to 1.7 for BlueJ
compatibility.
------------------------------------------------------------------------
r54 | jgillham | 2015-02-10 13:24:08 -0700 (Tue, 10 Feb 2015) | 1 line

Sped up the build building the jar directly from the JGit jar.
------------------------------------------------------------------------
r53 | jgillham | 2015-02-10 13:15:07 -0700 (Tue, 10 Feb 2015) | 1 line

Documented and cleaned code using Netbeans' suggestions.
------------------------------------------------------------------------
r52 | jgillham | 2015-02-10 12:34:34 -0700 (Tue, 10 Feb 2015) | 1 line

Renamed Git package to Helpers for naming coherency.
------------------------------------------------------------------------
r51 | jgillham | 2015-02-10 12:34:30 -0700 (Tue, 10 Feb 2015) | 1 line

Renamed Git package to Helpers for naming coherency.
------------------------------------------------------------------------
r50 | jgillham | 2015-02-09 13:46:43 -0700 (Mon, 09 Feb 2015) | 1 line

Prepared project for demo by removing unnecessary features and preventing
problems.
------------------------------------------------------------------------
r49 | jgillham | 2015-02-09 08:49:03 -0700 (Mon, 09 Feb 2015) | 1 line

Ignoring autogenerated folders.
```

```
------------------------------------------------------------------------
r48 | jgillham | 2015-02-09 08:39:43 -0700 (Mon, 09 Feb 2015) | 1 line

Removed unnecessary code and cleaned design.
------------------------------------------------------------------------
r47 | jgillham | 2015-02-07 21:44:13 -0700 (Sat, 07 Feb 2015) | 1 line

Uploaded meeting notes.
------------------------------------------------------------------------
r46 | mroman | 2015-02-07 16:36:39 -0700 (Sat, 07 Feb 2015) | 1 line

Added JavaDoc to files. Created BlueJHelper class for BlueJ related methods
------------------------------------------------------------------------
r45 | jgillham | 2015-02-06 16:12:42 -0700 (Fri, 06 Feb 2015) | 1 line

Refactored design. Seperating Git functionality from menu functionality.
------------------------------------------------------------------------
r44 | jgillham | 2015-02-06 16:12:38 -0700 (Fri, 06 Feb 2015) | 1 line

Refactored design. Seperating Git functionality from menu functionality.
------------------------------------------------------------------------
r43 | mroman | 2015-02-06 15:17:19 -0700 (Fri, 06 Feb 2015) | 1 line

Implemented GitCloneAction and made simple test case for it
------------------------------------------------------------------------
r42 | mroman | 2015-02-06 11:48:50 -0700 (Fri, 06 Feb 2015) | 1 line

Created and Implemented GitPushAction. Created test case for push command.
------------------------------------------------------------------------
r41 | mroman | 2015-02-05 17:00:43 -0700 (Thu, 05 Feb 2015) | 1 line

Refactored and moved some methods to a new GitHelper class
------------------------------------------------------------------------
r40 | jgillham | 2015-02-04 16:10:14 -0700 (Wed, 04 Feb 2015) | 1 line

Implemented the Git commit feature.
------------------------------------------------------------------------
r39 | nwitt | 2015-02-04 11:27:55 -0700 (Wed, 04 Feb 2015) | 1 line

updated url doc
------------------------------------------------------------------------
r38 | nwitt | 2015-02-04 11:23:57 -0700 (Wed, 04 Feb 2015) | 1 line

Added simple editor Test that prints "Moved!" in the debug file when the cursor
moves in the BlueJ editor. To install replace the "bluejeditor.jar" file in
blueJ's lib folder with the one in this directory.
------------------------------------------------------------------------
r37 | jgillham | 2015-02-03 18:03:15 -0700 (Tue, 03 Feb 2015) | 1 line

Created tests for Git Commit and Init.
------------------------------------------------------------------------
r36 | mroman | 2015-02-03 17:50:17 -0700 (Tue, 03 Feb 2015) | 1 line

Refactored GitExtension class and created new packages and classes
------------------------------------------------------------------------
r35 | mroman | 2015-02-03 16:22:58 -0700 (Tue, 03 Feb 2015) | 1 line

JUnit Testing class for GitExtension
```

```
------------------------------------------------------------------------
r34 | jgillham | 2015-02-03 16:11:46 -0700 (Tue, 03 Feb 2015) | 1 line

Including the JUnit and hamcrest jars.
------------------------------------------------------------------------
r33 | jgillham | 2015-02-03 16:00:36 -0700 (Tue, 03 Feb 2015) | 1 line

Added Unit Testing to the build.
------------------------------------------------------------------------
r32 | jgillham | 2015-02-03 15:46:19 -0700 (Tue, 03 Feb 2015) | 1 line

Added unit test frame work.
------------------------------------------------------------------------
r31 | mroman | 2015-02-03 15:40:16 -0700 (Tue, 03 Feb 2015) | 1 line

added test folder and test class
------------------------------------------------------------------------
r30 | jgillham | 2015-02-03 14:39:25 -0700 (Tue, 03 Feb 2015) | 1 line

Ignoring folders with *.classes.
------------------------------------------------------------------------
r29 | jgillham | 2015-02-03 14:08:44 -0700 (Tue, 03 Feb 2015) | 1 line

Including JGit classes into the extension jar.
------------------------------------------------------------------------
r28 | jgillham | 2015-02-03 00:51:25 -0700 (Tue, 03 Feb 2015) | 1 line

Added Git-Extension work from class session.
------------------------------------------------------------------------
r27 | whuang | 2015-02-02 15:25:43 -0700 (Mon, 02 Feb 2015) | 5 lines

Update to the ChirpEx.zip deliverable.

The ChirpEx.zip file is what we plan to deliver to our users.

Contains the Chirp.wav file, ChirpEx.jar extension, and a README.txt with
installation instructions.
------------------------------------------------------------------------
r26 | whuang | 2015-02-02 15:17:48 -0700 (Mon, 02 Feb 2015) | 1 line


------------------------------------------------------------------------
r25 | jgillham | 2015-02-01 15:06:25 -0700 (Sun, 01 Feb 2015) | 1 line

Added the jar into the project lib for convenience.
------------------------------------------------------------------------
r24 | jgillham | 2015-02-01 15:02:15 -0700 (Sun, 01 Feb 2015) | 1 line

Cleaned up JGit example.
------------------------------------------------------------------------
r23 | jgillham | 2015-01-31 17:02:08 -0700 (Sat, 31 Jan 2015) | 1 line

Added experiment that tests JGit.
------------------------------------------------------------------------
r22 | jgillham | 2015-01-31 16:57:27 -0700 (Sat, 31 Jan 2015) | 1 line

Created tasks for User Story 1.
------------------------------------------------------------------------
```

```
r21 | jgillham | 2015-01-31 16:19:31 -0700 (Sat, 31 Jan 2015) | 1 line

Finished the user stories for the release planning stage.
------------------------------------------------------------------------
r20 | whuang | 2015-01-31 16:09:58 -0700 (Sat, 31 Jan 2015) | 1 line

Updated source code and jar file. Removed pop up on compile.
------------------------------------------------------------------------
r19 | tmacari | 2015-01-31 16:01:41 -0700 (Sat, 31 Jan 2015) | 1 line

Added a Sample Project in which to test BlueJ SVN functionality.
------------------------------------------------------------------------
r18 | jgillham | 2015-01-31 15:28:26 -0700 (Sat, 31 Jan 2015) | 1 line

Added the notes from team meeting for 01/31/2015.
------------------------------------------------------------------------
r17 | dtran | 2015-01-31 14:05:47 -0700 (Sat, 31 Jan 2015) | 1 line

Created first draft of policies document
------------------------------------------------------------------------
r16 | dtran | 2015-01-29 22:50:26 -0700 (Thu, 29 Jan 2015) | 1 line

Added information addressing default library paths on different OSes.
------------------------------------------------------------------------
r15 | dtran | 2015-01-29 22:25:47 -0700 (Thu, 29 Jan 2015) | 1 line

Fixed directory backslash to forward slash, reuploaded resulting jar
------------------------------------------------------------------------
r14 | nwitt | 2015-01-29 19:59:27 -0700 (Thu, 29 Jan 2015) | 1 line

Added a test extension to test if this kind of extension is system independent.
------------------------------------------------------------------------
r13 | dtran | 2015-01-29 19:54:19 -0700 (Thu, 29 Jan 2015) | 1 line

Added Exploration Summary file, details on results of Project Chirp
------------------------------------------------------------------------
r12 | nwitt | 2015-01-29 16:02:33 -0700 (Thu, 29 Jan 2015) | 1 line

Added file chooser to simple chirp.
------------------------------------------------------------------------
r11 | nwitt | 2015-01-29 15:13:06 -0700 (Thu, 29 Jan 2015) | 1 line

added test class
------------------------------------------------------------------------
r10 | nwitt | 2015-01-29 14:50:20 -0700 (Thu, 29 Jan 2015) | 1 line

Got audio to work
------------------------------------------------------------------------
r9 | whuang | 2015-01-29 14:34:28 -0700 (Thu, 29 Jan 2015) | 1 line

Adding chirp sound effect
------------------------------------------------------------------------
r8 | whuang | 2015-01-29 14:24:32 -0700 (Thu, 29 Jan 2015) | 1 line

Updated source code
------------------------------------------------------------------------
r7 | whuang | 2015-01-29 14:17:05 -0700 (Thu, 29 Jan 2015) | 1 line
```

```
First Iteration - A pop up shows up on successful compile.
------------------------------------------------------------------------
r6 | mroman | 2015-01-29 13:11:18 -0700 (Thu, 29 Jan 2015) | 1 line

Created a documents folder withing Github and added a User Stories text file.
------------------------------------------------------------------------
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

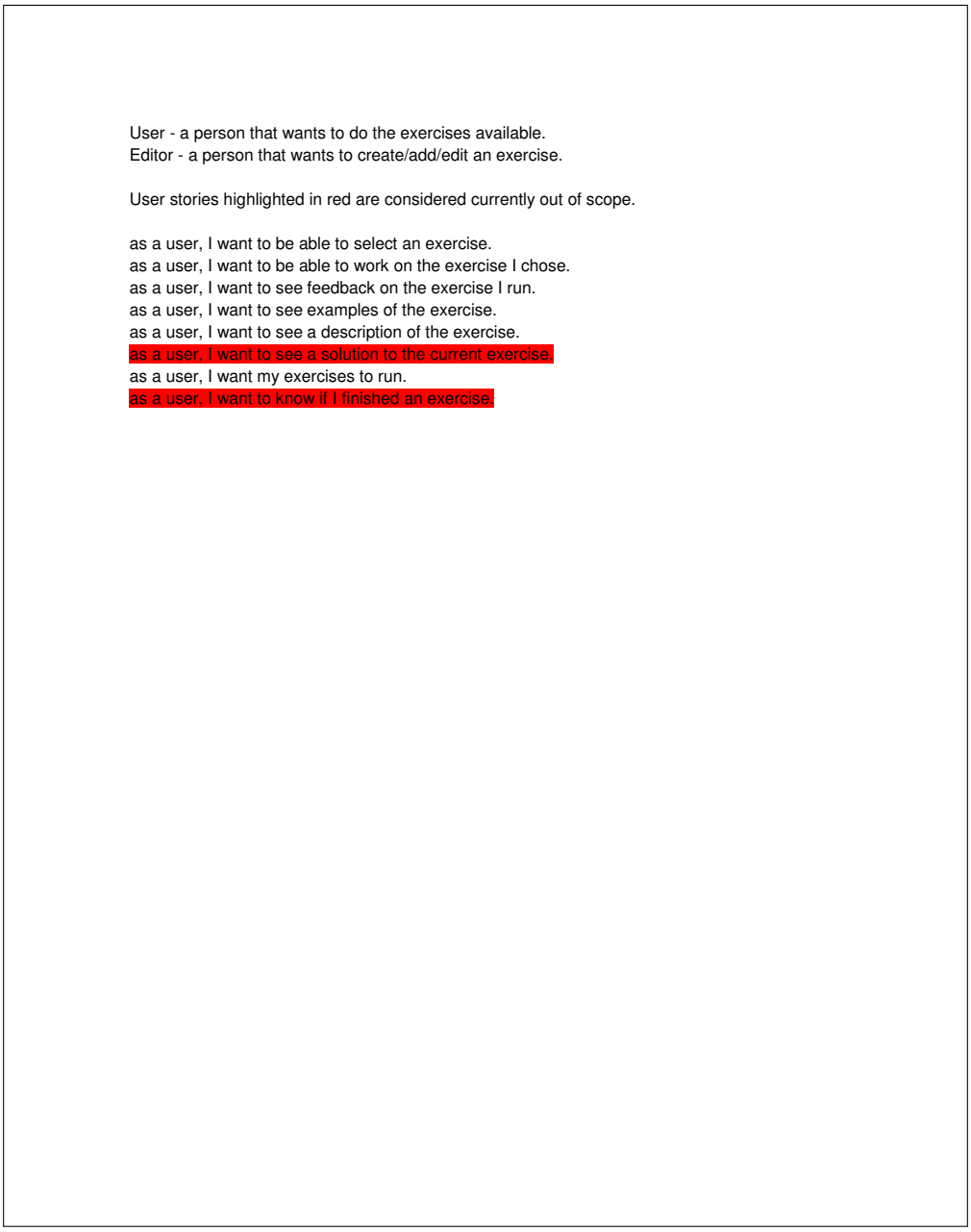
---

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

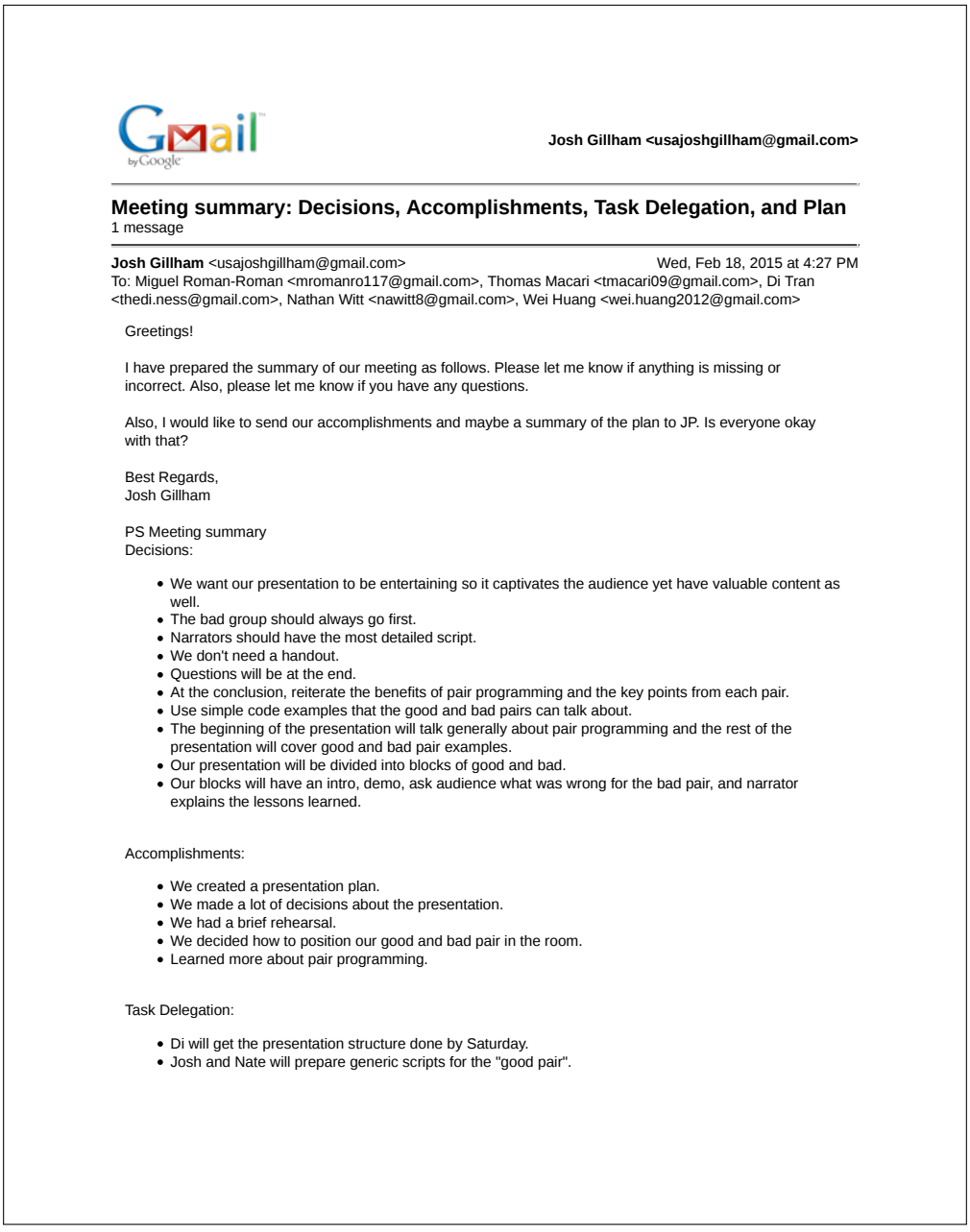
Description This artifact is a set of user stories that the team came up with. We created user stories to serve as requirements for the project. The user stories helped the team scope down the extensions functionalities to a concrete list of scenarios, and overall these stories allowed the team to share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

Josh Gillham <usajoshgillham@gmail.com>

In class meeting summary: Decisions, Delegation, Plan

1 message

Josh Gillham <usajoshgillham@gmail.com> Mon, Mar 2, 2015 at 4:26 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 realized that there is a gap in our plan, that is, we did not plan what we will work on during class on Wednesday. There our two possibilities that I see 1) we could do more design work 2) we could hash out the next presentation. Please let me know what you think.

Best Regards,
Josh Gillham

PS

Decisions:

- Final decision: we will be going with the coding bat idea that JP liked.
- We decided that as JP is our primary stake holder, we will place the most value on what he says about the ideas we choose.
- We would also like the students to be proud of what we make.

Delegation:

- Miguel, Wei will be working on the "add" function and will finish by class on Wednesday.
- Josh, Thomas will be working on the "do" function and will finish by class on Wednesday.

Plan:

- Wednesday before classes we will finish the prototypes.
- Wednesday after class we will ask students what they think of the prototypes.

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

In class meeting summary: Accomplishments, Todo, Todo Unassigned, Delegation, Current Status, Lessons Learned.

1 message

Josh Gillham <usajoshgillham@gmail.com> Wed, Mar 4, 2015 at 4:24 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 our summary post script. Please let me know if you have any questions.

Best Regards,
Josh Gillham

PS Summary

Accomplished:

- Created project definition, users stories, and entities.

Todo: Architecture, implementation, loading exercises, presentation.

Todo Unassigned: context diagram.

Delegation:

- Di, Nate will be coming up with the architecture by class on Monday.
- Wei, Miguel will be working the implementation for the GUI.
- Thomas is researching how tests could be loaded from a file.
- Josh will be working the presentation drafts which will include: a title page, TOC, and take home message.
- Josh will also float around when needed.

Current Status: Iteration one.

Lessons learned

- Pairs are good for creating ideas because people don't have to worry about their ideas getting disapproved.
- Meetings are good for invalidating ideas because there is so much transparency.
- There may be a feedback loop whereby first people pair off to create then come together to share and invalidate.

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

Meeting summary: Accomplishments, Monday's agenda
1 message

Josh Gillham <usajoshgillham@gmail.com> Sat, Mar 7, 2015 at 3:22 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 the meeting summary post script. I will create a separate status update draft for JP whereas this message is intended for internal communication. Please let me know if you have any comments or questions.

Best Regards,
Josh Gillham

PS Meeting summary

Accomplishments:

- Each pair updated the group with their findings.
- Created a status update.
- Each person choose their top 2 favorite presentations and presented 2 messages.
- We created an agenda for Monday.

Monday's agenda:

1. Miguel gives the status update.
2. Discuss the architecture that Di and Wei created.
3. Talk about the presentation and possibly choose one idea.
4. Wei will report on his findings about JavaFX.
5. Do the release planning game.
6. Do the iteration planning game.

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

In class meeting summary: JP Comments, Accomplishments, Delegation, and Decisions

1 message

Josh Gillham <usajoshgillham@gmail.com> Mon, Mar 16, 2015 at 4:30 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 include the meeting summary post script. The new meeting notes are up on SVN. Please let me know if you have any questions or comments.

Best Regards,
Josh Gillham

JP Comments:

- Our process model should specify only a small subset of characteristics we want and he will hold us accountable to that standard.
- For this, he recommends we find what we do now that early CS students would not do.
- He is concerned that a process model too formal might be a leach on our overhead.
- Our SVN structure does not account for versioning.

Accomplishments:

- Demoed XML parser.
- Demoed JUnit runner.
- Walked through process model demo with JP.
- Gave status report.

Delegation:

- We did not assign formal tasks but we did commit to improving our existing demos and extension source code.
- Wei and Miguel will be working on the source.
- Thomas and Nate will improve their demos.
- Di will be trying to simplify the process model by boiling it down to what we REALLY do.
- I (Josh) will work on the presentations and I might also think about how to boil down the process model.

Decisions:

- During the break, it is up to you how much you want to work, but, if you decide to work then specify what you are working on.

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

In class meeting summary: Saturday's agenda, Decisions, and Delegation

1 message

Josh Gillham <usajoshgillham@gmail.com> Wed, Mar 18, 2015 at 4:23 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 the meeting summary post script. Please let me know you have any questions.

Best Regards,
Josh Gillham

PS Meeting Summary

Saturday's agenda:

- Roundtable Report -- everybody gives their status and what they have accomplished.
- Presentation Practice -- We knock out one presentation as a group.

Decisions:

- We will be making a product that produces source code and checks the students answers.

Delegation:

- Di, Josh will be working on the presentation.
- Nate, Thomas, Miguel, and Wei will do coding.
- Thomas will upload his XML code by Saturday.
- Nate will upload his revised code sometime soon.
-

Josh Gillham <usajoshgillham@gmail.com>

In class meeting summary: Delegation, Presentation Reflection, Targets, JP's comments

1 message

Josh Gillham <usajoshgillham@gmail.com> Wed, Apr 1, 2015 at 5:01 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!

Here is a summary from today. Please let me know if I missed something. Also, lets continue this thread to discuss design decisions.

Best Regards,
Josh Gillham

PS In class meeting summary:

Notes:

- Nate and Thomas cannot make the Saturday meeting.

Targets:

- The GUI loader button.
- The backend loader and focused on exercise directories.

Delegation:

- Miguel will commit his code.
- Nate will try and integrate his runner.
- Nate may do some thinking and work on the design.
- Thomas will make several informal reports:
  - Recommend a commit policy.
  - Recommend a improvement to the SVN structure.
- Everybody learns more about the project.

Presentation Reflection:

- Practice on note cards.
- More rehearsals.
- Pick a topic that we have more experience in.
- Try one person standing and the rest sit down.
- Don't go off topic or into extra information during questions.
- More handout fields like Group and Individual names and maybe a TOC.

JP's comments:

Gmail by Google

Josh Gillham <usajoshgillham@gmail.com>

In class meeting summary

7 messages

Josh Gillham <usajoshgillham@gmail.com> Wed, Apr 8, 2015 at 4:22 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!

This will be a thread to reply with any thoughts that should not be forgotten.

Best Regards,
Josh Gillham

PS Meeting summary

Decisions

- Di would like us to try using Skype for the next meeting. My username is: usajgillham

---

Wei Huang <wei.huang2012@gmail.com> Wed, Apr 8, 2015 at 9:33 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>

Wei* would like us to try using skype for the next meeting. Haha
I sent a contact request to you.
My username is Llria.
Typing in weimingden should also find this account.
[Quoted text hidden]

Josh Gillham <usajoshgillham@gmail.com> Wed, Apr 8, 2015 at 10:38 PM
To: Wei Huang <wei.huang2012@gmail.com>
Cc: N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>

Wemmingden?

[Quoted text hidden]

Wei Huang <wei.huang2012@gmail.com> Thu, Apr 9, 2015 at 4:46 AM
To: Josh Gillham <usajoshgillham@gmail.com>
Cc: N Witt <nawitt8@gmail.com>, Thomas Macari <tmacari09@gmail.com>, Di Tran <thedi.ness@gmail.com>, Miguel Roman-Roman <mromanro117@gmail.com>

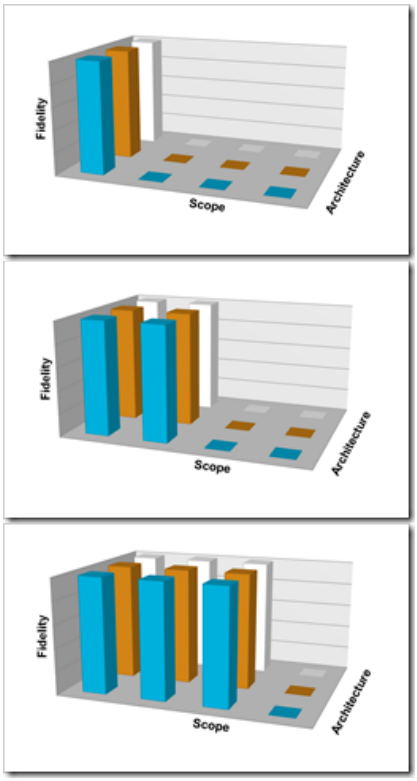
Software Development Life Cycle - Iterative and Incremental Model

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)

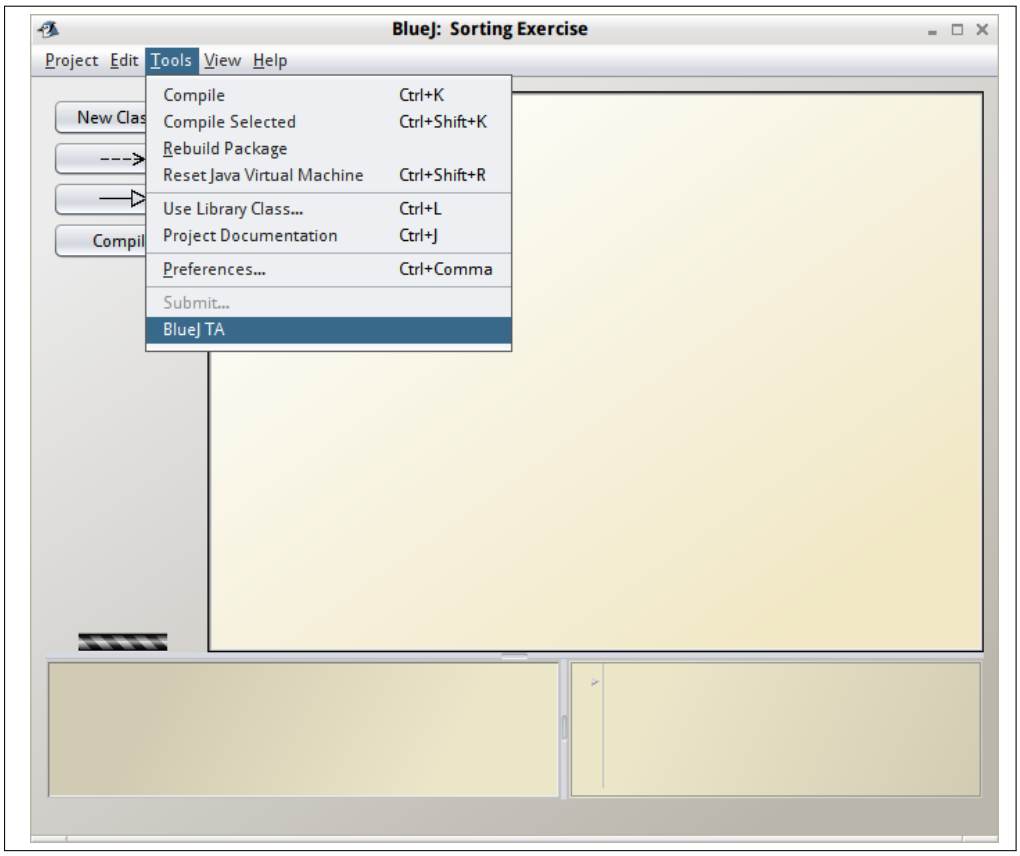


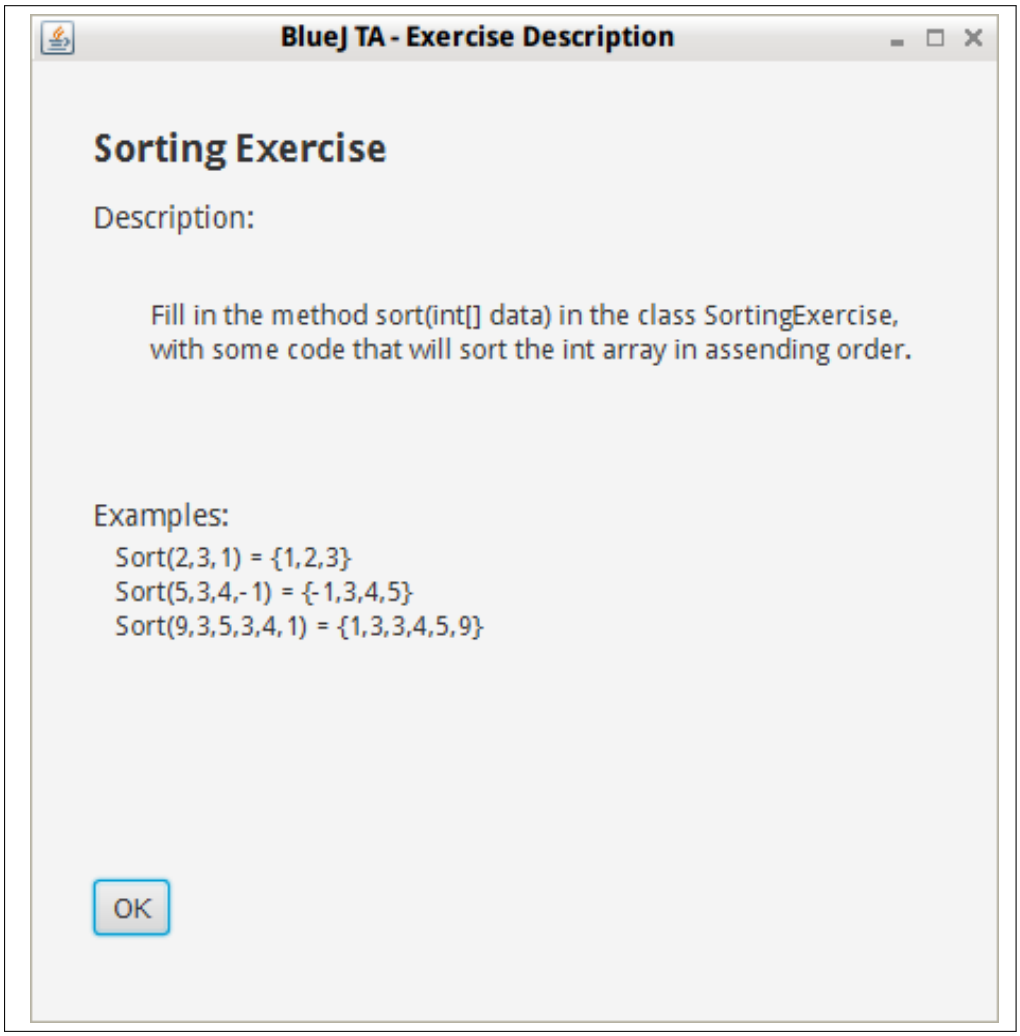
Product

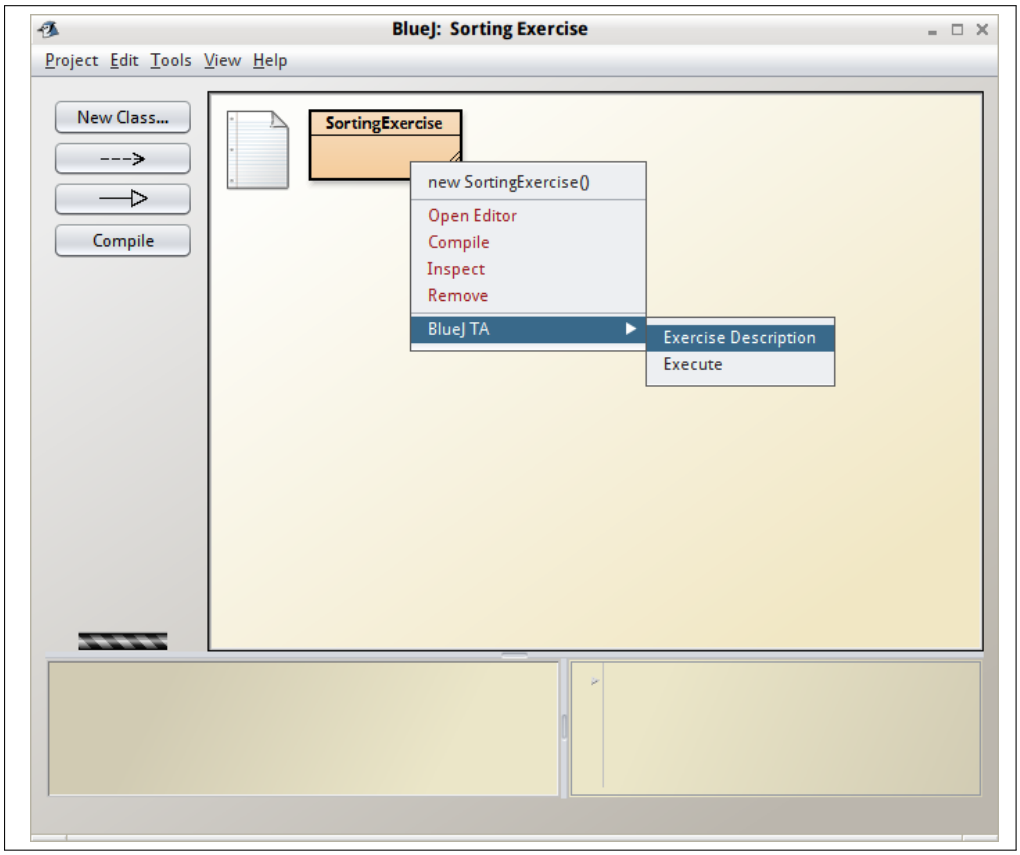
Summary Our product, BlueJ TA, is a BlueJ extension that aids in the learning process by providing programming exercises for students to complete and verify in real-time. The following section describes the elements that as a whole compose our product. Other than the BlueJ extension menus, our project 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.

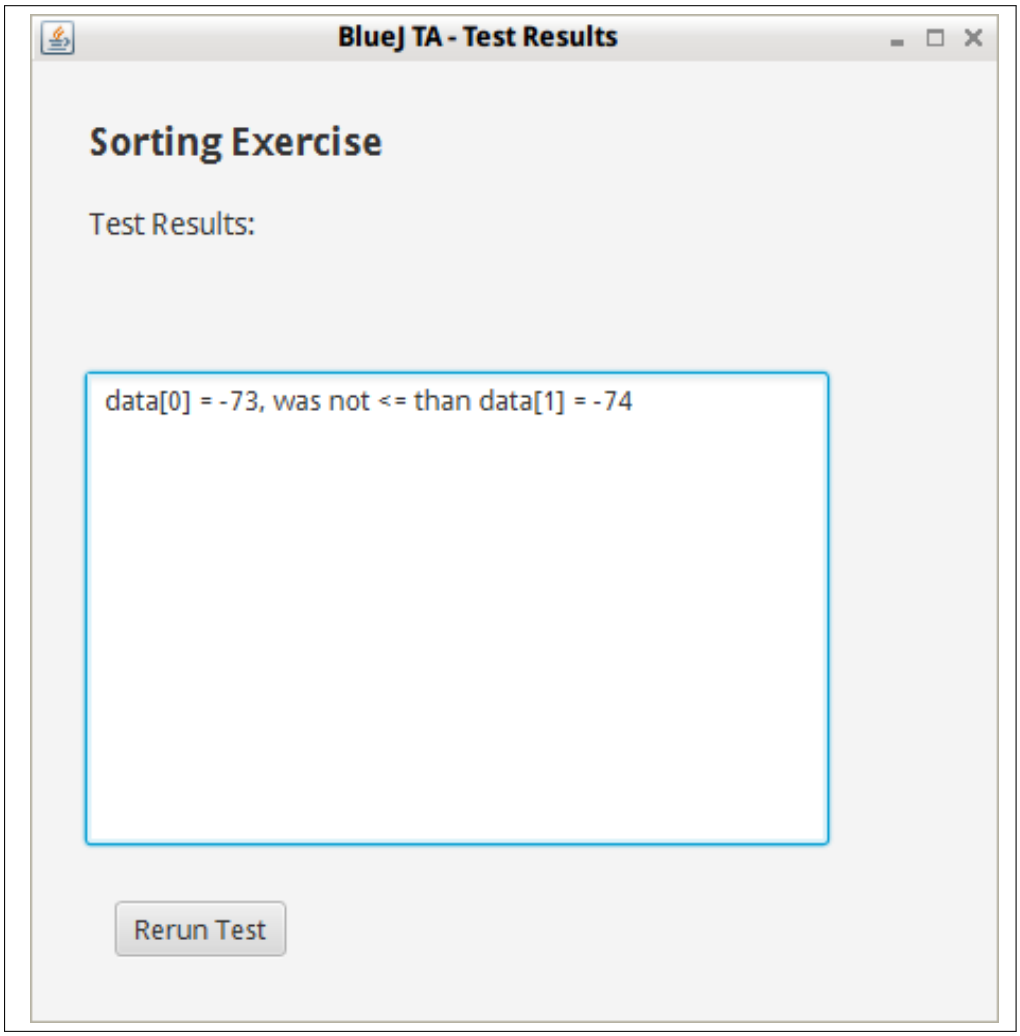
BlueJ TA

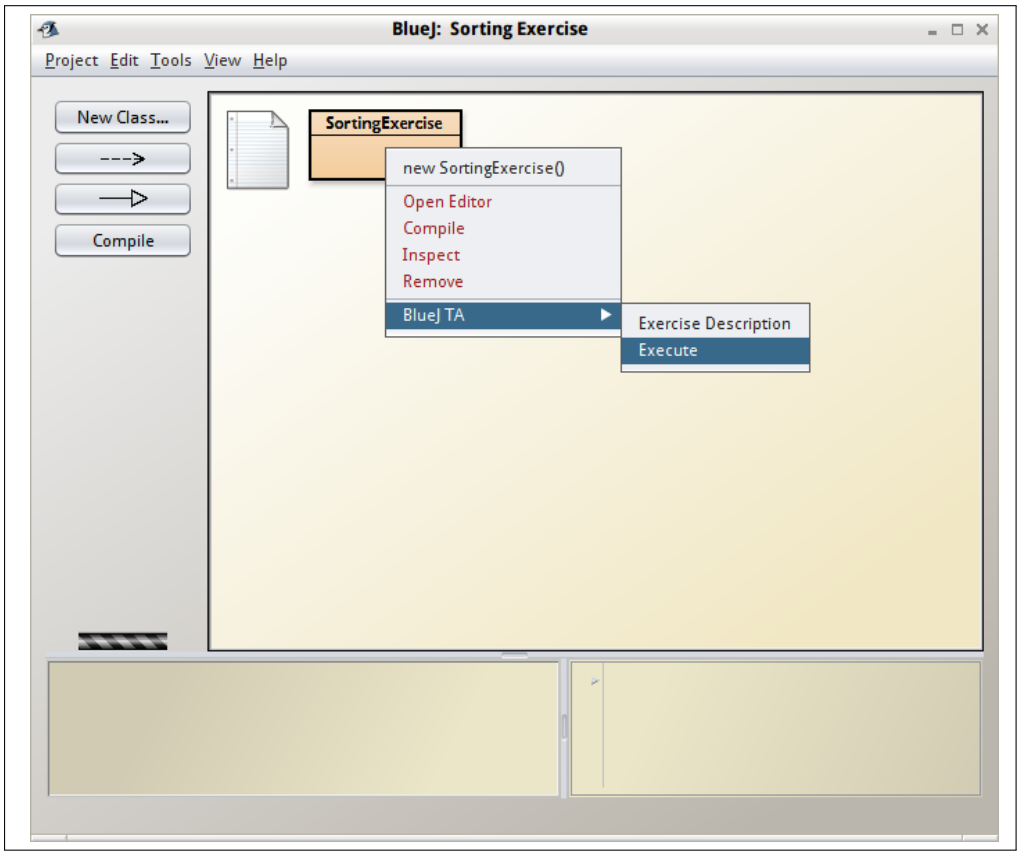
Description The following artifact is the BlueJ TA extension our team developed. This product is composed of three main modules: the graphical user interfaces, the XML parser, and the JUnitRunner. This program was the outcome of a 16 week project for the CS 4260 course. This program is in a fully functional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

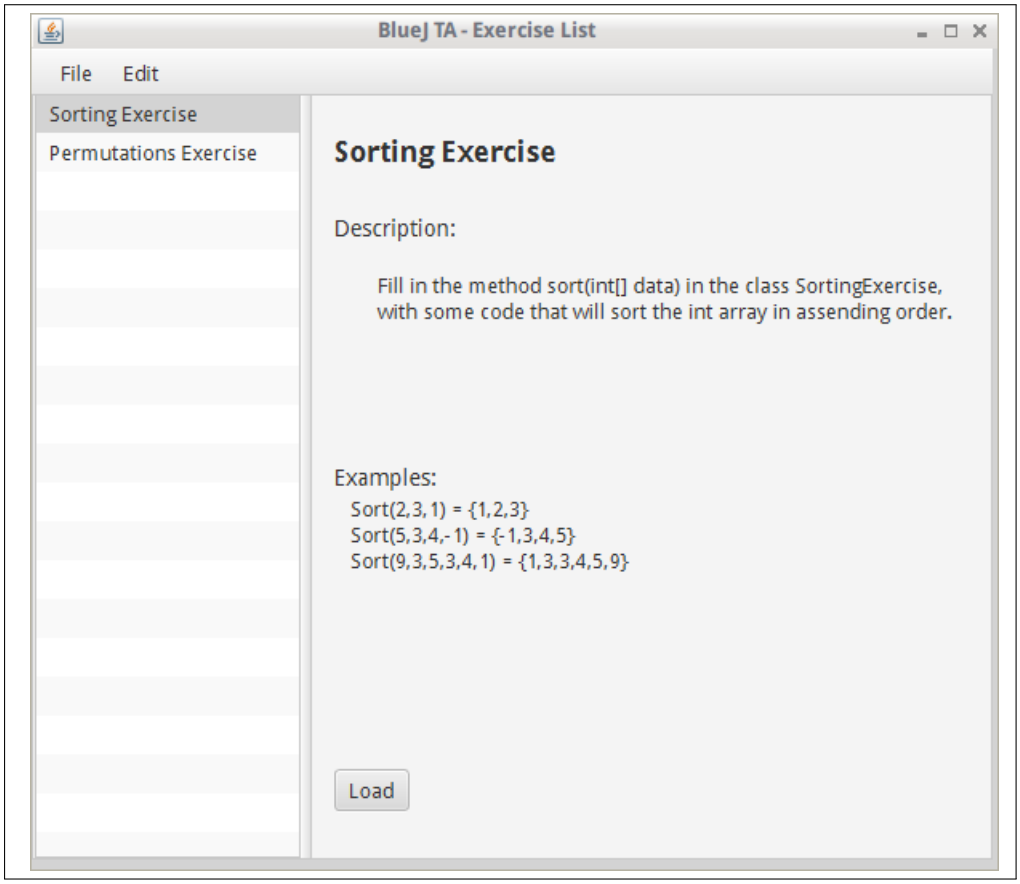


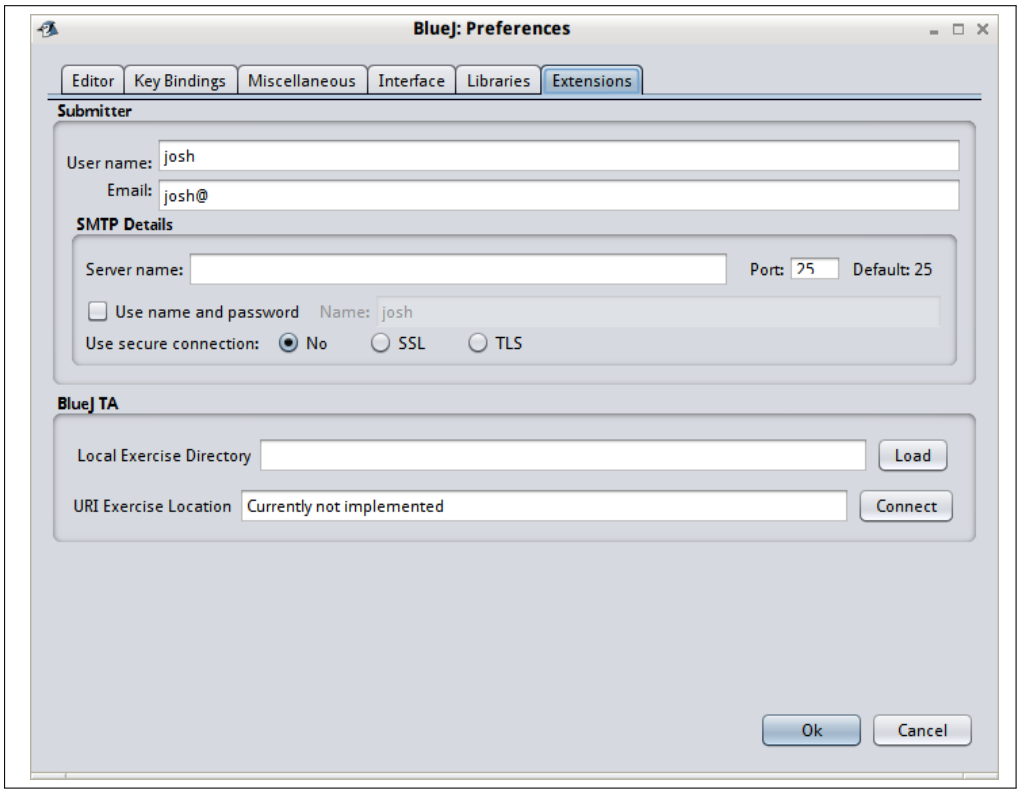

BlueJ TA - Exercise Description
Sorting Exercise
Description:
Fill in the method sort(int[] data) in the class SortingExercise,
with some code that will sort the int array in assending order.
Examples:
Sort(2,3,1) = {1,2,3}
Sort(5,3,4,-1) = {-1,3,4,5}
Sort(9,3,5,3,4,1) = {1,3,3,4,5,9}
OK

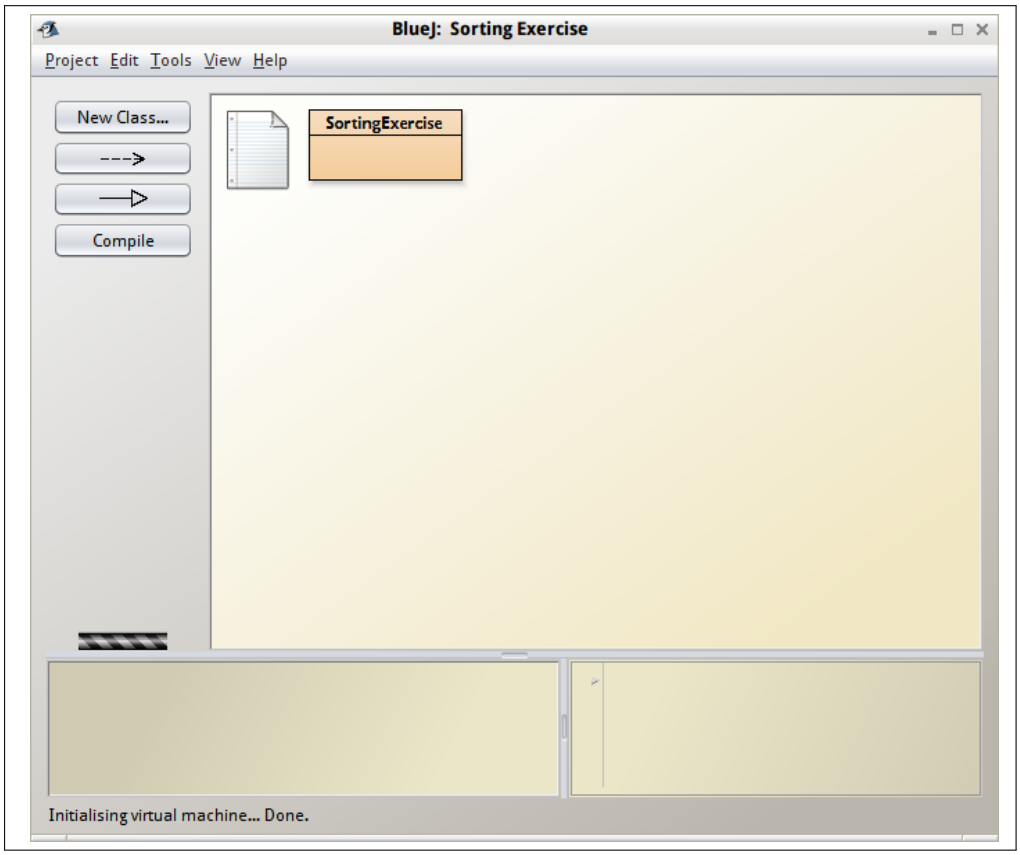

BlueJ: Sorting Exercise
Project Edit Tools View Help
New Class...
Compile
SortingExercise
new SortingExercise()
Open Editor
Compile
Inspect
Remove
BlueJ TA
Exercise Description
Execute


BlueJ TA - Test Results
Sorting Exercise
Test Results:
data[0] = -73, was not <= than data[1] = -74
Rerun Test


BlueJ: Sorting Exercise
Project Edit Tools View Help
New Class...
Compile
SortingExercise
new SortingExercise()
Open Editor
Compile
Inspect
Remove
BlueJ TA
Exercise Description
Execute


BlueJ TA - Exercise List
File Edit
Sorting Exercise
Permutations Exercise
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}
Load


BlueJ: Preferences
Editor
Key Bindings
Miscellaneous
Interface
Libraries
Extensions
Submitter
User name:
josh
Email:
josh@
SMTP Details
Server name:
Port:
25
Default: 25
Use name and password
Name:
josh
Use secure connection:
No
SSL
TLS
BlueJ TA
Local Exercise Directory
Load
URI Exercise Location
Currently not implemented
Connect
Ok
Cancel


BlueJ: Sorting Exercise
Project
Edit
Tools
View
Help
New Class...
Compile
SortingExercise
Initialising virtual machine... Done.

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
 */
@XmlRootElement
public class Exercise
{

    //the title of the exercise and project when laucnched
    private String title;

    //the description of the exercise depicting what is required of the user and wha
    private String description;

    //a hint that could help the user if they are stuck
    private String hint;

    //a sample solution to this exercise
    private String sampleAnswer;

    //the class name of the java file the user starts with
    private String javaName;

    //the contents of the java file the user starts with
    private String javaCode;

    //the class name of the junit java file that will be used to run the test
    private String junitName;

    //the contents of the junit java file that will be used to run the test
    private String junitCode;

    //a set of examples
```

```
private List<String> example;

//the JUnit test file this exercise will use to execute test with.
private File testFile;

//the BlueJ package this exercise is associated with.
private BPackage exercisePkg;

/**
 * Constructor
 */
private Exercise()
{
}

/**
 * Returns the name of the Java file for this exercise.
 *
 * @return  the name of the Java file
 */
@XmlElement
public String getJavaName()
{
    return javaName;
}

/**
 * Sets the Java file name for this exercise.
 *
 * @param   the name of the Java file
 */
public void setJavaName(String javaName)
{
    this.javaName = javaName;
}

/**
 * Returns the Java code for the exercise file of this exercise.
 *
 * @return  the Java code
 */
@XmlElement
public String getJavaCode()
{
    return javaCode;
}

/**
 * Sets the Java code of the exercise file of this exercise.
 *
 * @param javaCode  the Java code
 */
public void setJavaCode(String javaCode)
{
    this.javaCode = javaCode;
}

/**
```

```
     * Returns the name of the JUnit Java file for this exercise.
     *
     * @return  the name of the JUnit Java file
     */
    @XmlElement
    public String getJunitName()
    {
        return junitName;
    }

    /**
     * Sets the name of the JUnit Java file associated with this exercise.
     *
     * @param   the name of the JUnit Java file
     */
    public void setJunitName(String junitName)
    {
        this.junitName = junitName;
    }

    /**
     * Returns the Java code of the JUnit file associated with this exercise.
     *
     * @return  the Java code for the JUnit file
     */
    @XmlElement
    public String getJunitCode()
    {
        return junitCode;
    }

    /**
     * Sets the Java code of the JUnit file associated with this exercise.
     *
     * @param junitCode the Java code for the JUnit file
     */
    public void setJunitCode(String junitCode)
    {
        this.junitCode = junitCode;
    }

    /**
     * Sets the list of examples for this exercise.
     *
     * @param exampleList   the List of examples
     */
    public void setExampleList(List<String> exampleList)
    {
        this.example = exampleList;
    }

    /**
     * Used to access the title of this exercise.
     *
     * @return  the title of this exercise
     */
    @XmlElement
    public String getTitle()
```

```
{
    return title;
}

/**
 * Used to access the description of this exercise.
 *
 * @return  the description of this exercise
 */
@XmlElement
public String getDescription()
{
    return description;
}

/**
 * Used to access the hint of this exercise.
 *
 * @return  the hint of this exercise
 */
@XmlElement
public String getHint()
{
    return hint;
}

/**
 * Used to access the sample answer of this exercise.
 *
 * @return  the sample answer of this exercise
 */
@XmlElement
public String getSampleAnswer()
{
    return sampleAnswer;
}

/**
 * Used to set the title of this exercise.
 *
 * @param title the title
 */
public void setTitle(String title)
{
    this.title = title;
}

/**
 * Used to set the description of this exercise.
 *
 * @param description   the description
 */
public void setDescription(String description)
{
    this.description = description;
}

/**
```

```
     * Used to set the hint of this exercise.
     *
     * @param hint  the hint
     */
    public void setHint(String hint)
    {
        this.hint = hint;
    }

    /**
     * Used to set the sample answer of this exercise.
     *
     * @param sampleAnswer  the sample answer
     */
    public void setSampleAnswer(String sampleAnswer)
    {
        this.sampleAnswer = sampleAnswer;
    }

    /**
     * Used to set the example list of this exercise.
     *
     * @param example   the example List
     */
    public void setExample(List<String> example)
    {
        this.example = example;
    }

    /**
     * Used to access the examples list of this exercise.
     *
     * @return  the List of examples
     */
    @XmlElementWrapper(name = "examples")
    @XmlElement(name = "example")
    public List<String> getExample()
    {
        return example;
    }

    /**
     * Launches this exercise by setting up the JUnit test and user project.
     */
    public void launch()
    {
        BlueJ bluej = StateManager.getBlueJ();

        //close any open projects with this exercise name
        FileUtil.closeProj(bluej, title);

        //sets up a temp directory and project for this exercise
        File exerciseTempDir = new File(StateManager.getTempDir(), title + "_Files")
        FileUtil.deleteDir(exerciseTempDir);
        exerciseTempDir.mkdir();
        File projDir = new File(exerciseTempDir, title);
        BProject bproj = bluej.newProject(projDir);
```

```
        Frame frame = null;
        try
        {
            BPackage bpkg = bproj.getPackage("");

            //locks the ui from repainting, to hide the junit class while it is in t
            frame = bpkg.getFrame();
            frame.getComponent(0).setVisible(false);

            //saves the users starting java files and the JUnit test files to the pr
            FileUtil.save(new File(projDir, javaName + ".java"), javaCode);
            FileUtil.save(new File(projDir, junitName + ".java"), junitCode);

            //create new BClasses from the files
            bpkg.newClass(javaName);
            BClass bTestClass = bpkg.newClass(junitName);

            //compile the files and wait till compilation finishes
            bpkg.compileAll(true);

            //copies the JUnit test class file out of the users project and remember
            testFile = new File(exerciseTempDir, junitName + ".class");
            FileUtil.copy(bTestClass.getClassFile(), testFile);

            //remembers this package to be used later in the execution of the junit
            exercisePkg = bpkg;

            //removes the JUnit test file from the project
            bTestClass.remove();
        } catch (Exception ex) {
            System.err.println("Error: " + ex);
        }
        if (frame != null)
        {
            //unlocks the ui to be repainted again
            frame.getComponent(0).setVisible(true);
            frame.toFront();
        }
    }

    /**
     * Runs this exercises JUnit test with the code within the users exercise
     * project. This will do this using BlueJ's user VM feature.
     *
     * @return the results of this exercise.
     */
    public String execute()
    {
        try
        {
            //save the needed files into the users project
            File userProjDir = exercisePkg.getDir();
            File projTestFile = new File(userProjDir, testFile.getName());
            FileUtil.copy(testFile, projTestFile);
            FileUtil.extractResource("runner/", "JRunner.class", userProjDir);

            //locks the ui from repainting, to hide the added classes while they are
            Frame frame = exercisePkg.getFrame();
```

```
            frame.getComponent(0).setVisible(false);

            //reload package to allow the added classes to be obtained
            exercisePkg.reload();
            BClass testClass = exercisePkg.getBClass(junitName);
            BClass runClass = exercisePkg.getBClass("JRunner");

            //reset the static runner flag in the JRunner class, so the last test do
            runClass.getMethod("resetRunning", null).invoke(null, null);

            //get the method to execute
            BMethod runMethod = runClass.getMethod("run", new Class<?>[]{String.clas

            //start up a remove thread that will remove the added classes and unlock
            new RemoveThread(testClass, runClass, frame).start();

            //execute the junit test
            Object results = runMethod.invoke(null, new Object[]{junitName});

            return results.toString();
        } catch (InvocationErrorException e)
        {
            return "Test Canceled by user.";
        } catch (Exception e)
        {
            e.printStackTrace();
            return e.toString();
        }
    }

    /**
     * Prints the various data that composes this exercise
     */
    public void printExercise()
    {
        System.out.println("Title: " + title);
        System.out.println("Description: " + description);
        System.out.println("Hint: " + hint);
        System.out.println("Sample Answer: " + sampleAnswer);
        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
     * 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.
     *
     * @param path the file to save
     * @param contents the contents to save into the file
     * @return if the save was successful.
     */
    public static boolean save(File path, String contents)
    {
        String[] lines = contents.split(System.lineSeparator());
        try {
            PrintWriter out = new PrintWriter(path);
            for (String line : lines)
            {
                out.println(line);
            }
            out.close();
            return true;
        } catch (Exception e)
        {
            return false;
        }
    }

    /**
     * Copies the contents of the source file into the destination file.
     *
     * @param source the source file
     * @param dest the destination file
     * @references
```

```
     * http://examples.javacodegeeks.com/core-java/io/file/4-ways-to-copy-file-in-ja
     */
    public static void copy(File source, File dest)
    {
        InputStream input = null;
        OutputStream output = null;
        try {
            input = new FileInputStream(source);
            output = new FileOutputStream(dest);
            transferData(input, output);
        } catch (Exception e) {
            System.err.println("Error: " + e);
        }
    }

    /**
     * Saves the given resource within this jar to the a file in the directory
     * specified.
     *
     * @param resLoc the directory within the jar to look for the file, relative
     * to this FileUtil class.
     * @param resName the name of the file to extract. This parameter will also
     * act as the name of the extracted file.
     * @param destDir the directory to save the file to.
     */
    public static void extractResource(String resLoc, String resName, File destDir)
    {
        try {
            InputStream input = FileUtil.class.getResourceAsStream(resLoc + resName)
            OutputStream output = new FileOutputStream(new File(destDir, resName));
            transferData(input, output);
        } catch (Exception e)
        {
            System.err.println("Error: " + e);
        }
    }

    /**
     * Transfers the data from the given InputStream to the given OutputStream
     *
     * @param input the InputStream to retrieve data from.
     * @param output the OutputStream to send the data to.
     */
    private static void transferData(InputStream input, OutputStream output) throws
    {
        byte[] buf = new byte[1024];
        int bytesRead;
        while ((bytesRead = input.read(buf)) > 0)
        {
            output.write(buf, 0, bytesRead);
        }
        input.close();
        output.close();
    }

    /**
     * Deletes the directory and all the contents within the directory.
     *
```

```
     * @param path the directory to delete.
     * @return if the directory was deleted successfully.
     * @references http://www.rgagnon.com/javadetails/java-0483.html
     */
    public static boolean deleteDir(File path)
    {
        if (path.exists()) {
            File[] files = path.listFiles();
            for (int i = 0; i < files.length; i++)
            {
                if (files[i].isDirectory())
                {
                    deleteDir(files[i]);
                } else
                {
                    files[i].delete();
                }
            }
        }
        return (path.delete());
    }

    /**
     * Closes the any open bluej projects that have the given name.
     *
     * @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
    private final BClass testClass, runClass;

    /**
     * Constructor
     *
     * @param testClass the JUnit test class to remove.
     * @param runClass the JRunner class to remove.
     * @param frame the frame to unlock after removal.
     */
    RemoveThread(BClass testClass, BClass runClass, Frame frame)
    {
        this.testClass = testClass;
        this.runClass = runClass;
        this.frame = frame;
    }

    /**
     * Remove the BClasses from the project and unlock the UI after the JRunners
     * running flag has been set to true.
     */
    @Override
    public void run()
    {
        try
        {
            BField runFlag = runClass.getField("running");
            BObject runRef = runClass.getConstructor(null).newInstance(null);
            while (!(Boolean) runFlag.getValue(runRef))
            {
                Thread.sleep(100);
            }
        } catch (Exception e)
        {
            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

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
 */
public class JRunner
{

    /**
     * A flag used to signal when its ok for the extension to remove
     * this class and the JUnit test class files from the exercise project.
     */
    public static boolean running = false;

    /**
     * Resets the static running flag for a future test to be run.
     */
    public static void resetRunning()
    {
        running = false;
    }

    /**
     * Executes the JUnit Test of a given JUnit class and returns the results in
```

```
     * the form of a string.
     *
     * @param testClass the name of the JUnit test class to execute.
     * @return the results of the test
     */
    public static String run(String testClass)
    {
        Class testC = null;
        try
        {
            testC = Class.forName(testClass);
        } catch (Exception e)
        {
            return e.getMessage();
        }

        //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

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

---

```
package Extension.BackEnd;

import java.awt.event.ActionEvent;
import javax.swing.SwingUtilities;
import java.util.ArrayList;
import java.io.File;
import org.junit.runner.Result;
import java.io.*;
import javax.swing.*;
import java.nio.file.Path;
import bluej.extensions.BlueJ;
import bluej.extensions.BProject;
import bluej.extensions.BPackage;
import bluej.extensions.BClass;
```

```
import java.util.Scanner;
import java.util.List;
import java.util.LinkedList;

import Extension.Main;
import Extension.GUI.*;

/**
 * The StateManager class controls the current state of the extension.
 * It holds all necessary information such as the GUI's and the currently chosen exe
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class StateManager {
    private static BlueJ blueJ;

    private static String exercisesLocation;
    private static ArrayList<Exercise> exercises;
    private static Exercise currentExercise;

    private static TestResultsGUI testResultsGUI;
    private static DescriptionGUI descriptionGUI;
    private static ExerciseListGUI exerciseListGUI;

    private static File tempDir, extDir;

    /**
     * Private constructor
     */
    private StateManager()
    {
    }

    /**
     * Used to set up this State Manager and extension.
     * Creates proper folders in the BlueJ config directory.
     * Parses the exercises from known location.
     *
     * @param bluej the BlueJ instance
     */
    public static void setup(BlueJ bluej)
    {
        blueJ = bluej;
        tempDir = new File(blueJ.getUserConfigDir(), "TA_Temp");
        FileUtil.deleteDir(tempDir);
        tempDir.mkdir();

        StateManager.parseExercises();
    }

    /**
     * Returns the BlueJ instance held
```

```
     *
     * @return  the BlueJ instance
     */
    public static BlueJ getBlueJ()
    {
        return blueJ;
    }

    /**
     * Returns a File for the location of the Temporary Directory created by this ex
     *
     * @return  the location of the Temporary Directory
     */
    public static File getTempDir()
    {
        return tempDir;
    }

    /**
     * Returns the current exercise chosen by the user.
     *
     * @return  the currently selected exercise
     */
    public static Exercise getCurrentExercise()
    {
        return currentExercise;
    }

    /**
     * Sets the current exercise chosen by the user.
     *
     * @param ex    the exercise
     */
    public static void setCurrentExercise(Exercise ex)
    {
        currentExercise = ex;
    }

    /**
     * Returns the String location of the exercises.
     *
     * @return  the path of the exercises
     */
    public static String getExercisesLocation()
    {
        return exercisesLocation;
    }

    /**
     * Sets the location of the exercises.
     *
     * @param location  the location of the exercises
     */
    public static void setExercisesLocation(String location)
    {
        exercisesLocation = location;
    }
```

```
/**
 * Returns the held instance of the TestResultGUI
 *
 * @return  the TestResultsGUI
 */
public static TestResultsGUI getTestResultsGUI()
{
    return testResultsGUI;
}

/**
 * Returns the held instance of the DescriptionGUI
 *
 * @return  the DescriptionGUI
 */
public static DescriptionGUI getDescriptionGUI()
{
    return descriptionGUI;
}

/**
 * Returns the held instance of the ExerciseListGUI
 *
 * @return the ExerciseListGUI
 */
public static ExerciseListGUI getExerciseListGUI()
{
    return exerciseListGUI;
}

/**
 * Sets the instance of TestResultsGUI
 *
 * @param gui   the TestResultsGUI
 */
public static void setTestResultsGUI(TestResultsGUI gui)
{
    testResultsGUI = gui;
}

/**
 * Sets the intance of DescriptionGUI
 *
 * @param gui   the DescriptionGUI
 */
public static void setDescriptionGUI(DescriptionGUI gui)
{
    descriptionGUI = gui;
}

/**
 * Sets the instance of ExerciseListGUI
 *
 * @param gui   the ExerciseListGUI
 */
public static void setExerciseListGUI(ExerciseListGUI gui)
{
    exerciseListGUI = gui;
```

```
}

/**
 * Initializes the exercise list.
 *
 * @return exercises  the list of exercises.
 */
public static ArrayList<Exercise> getExerciseList()
{
    return exercises;
}

/**
 * Used to show an error message.
 * Displays a JOptionPane with given message.
 *
 * @param msg   the message to display
 */
public static void showErrorMessage(String msg)
{
    JOptionPane.showMessageDialog(null, msg, "Error", JOptionPane.ERROR_MESSAGE)
}

/**
 * Parses the exercises in the location specified and populates the exercise lis
 */
public static void parseExercises()
{
    exercises = new ArrayList<Exercise>();
    System.out.println( exercisesLocation );
    if (exercisesLocation == null || exercisesLocation.isEmpty())
    {
        String listDir = "/Extension/exercises/exerciseList.lst";
        Scanner in = new Scanner(Main.class.getResourceAsStream(listDir));
        XMLReader reader = new XMLReader();
        while (in.hasNextLine())
        {
            String ex = in.nextLine();
            System.out.println("ex: " + ex);
            exercises.add(reader.readExercise(Main.class.getResourceAsStream(ex)
        }
    }else
    {
        File directory = new File(exercisesLocation);
        System.out.println(directory.toPath());
        File[] filesInDir = directory.listFiles();
        if(filesInDir != null){
            XMLReader reader = new XMLReader();
            for(int i = 0; i < filesInDir.length; i++)
            {
                if(filesInDir[i].isFile() && !filesInDir[i].isHidden())
                {
                    System.out.println("File: " + filesInDir[i].toPath().toStrin
                    exercises.add(reader.readExercise(filesInDir[i].toPath().toS
                }
            }
        }
    }
```

```
        System.out.println("Size: " + exercises.size());
    }

    /**
     * Creates the GUI showing the JUnit tests for the exercise.
     */
    public static void createTestResultsGUI()
    {
        if (testResultsGUI == null)
        {
            SwingUtilities.invokeLater(new Runnable()
            {
                @Override
                public void run()
                {
                    testResultsGUI = new TestResultsGUI();
                }
            });
        } else if(!testResultsGUI.isVisible())
        {
            testResultsGUI.setVisible(true);
        }
    }

    /**
     * Creates the GUI for the description of the exercise.
     */
    public static void createDescriptionGUI()
    {
        if (descriptionGUI == null)
        {
            SwingUtilities.invokeLater(new Runnable()
            {
                @Override
                public void run()
                {
                    descriptionGUI = new DescriptionGUI();
                }
            });
        } else if(!descriptionGUI.isVisible())
        {
            descriptionGUI.setVisible(true);
        }
    }

    /**
     * Creates the GUI for picking an exercise.
     */
    public static void createExerciseListGUI()
    {
        if (exerciseListGUI == null)
        {
            SwingUtilities.invokeLater(new Runnable()
            {
                @Override
                public void run() {
                    exerciseListGUI = new ExerciseListGUI();
                }
```

```
            });
        } else if(!exerciseListGUI.isVisible())
        {
            exerciseListGUI.setVisible(true);
        }
    }

    /**
     * Disposes the ExerciseListGUI
     */
    public static void disposeExerciseGUI()
    {
        if( exerciseListGUI != null)
        {
            exerciseListGUI.setVisible(false);
            exerciseListGUI.dispose();
            exerciseListGUI = null;
        }
    }

    /**
     * Disposes the DescriptionGUI
     */
    public static void disposeDescriptionGUI()
    {
        if( descriptionGUI != null )
        {
            descriptionGUI.setVisible(false);
            descriptionGUI.dispose();
            descriptionGUI = null;
        }
    }

    /**
     * Disposes the TestResultsGUI
     */
    public static void disposeTestResultsGUI()
    {
        if( testResultsGUI != null )
        {
            testResultsGUI.setVisible(false);
            testResultsGUI.dispose();
            testResultsGUI = null;
        }
    }
}
```

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

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 objec
 * 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
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class XMLReader
{
    private JAXBContext context;
    private Unmarshaller um;
    private Exercise ex;

    /**
     * Constructor
     */
    public XMLReader()
    {
    }

    /**
     * Reads an XML file that has been converted into an InputStream
     *
     * @param is    the XML file as InputStream
     * @return  the Exercise object created from the XML file.
     */
    public Exercise readExercise(InputStream is)
    {
        ex = null;
        try
        {
            context=JAXBContext.newInstance(Exercise.class);
            um=context.createUnmarshaller();
            ex=(Exercise) um.unmarshal(new InputStreamReader( is ) );
        } catch (JAXBException ex)
        {
            ex.printStackTrace();
        }
        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

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

---

```
package Extension.GUI;

import javafx.application.Platform;
import javafx.embed.swing.JFXPanel;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import java.awt.event.WindowListener;
import java.awt.event.WindowEvent;

import static javax.swing.JFrame.EXIT_ON_CLOSE;

import Extension.BackEnd.*;

/**
 * The Graphical User Interface used to show the description of the exercises the us
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class DescriptionGUI extends JFrame implements WindowListener
{
    private JFXPanel panel;

    /**
     * Creates the GUI
     */
    public DescriptionGUI()
    {
```

```
        this.setTitle("BlueJ TA - Exercise Description");
        this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        this.setSize(457, 471);
        this.addWindowListener(this);
        this.setResizable(false);
        this.setLocationRelativeTo(null);
        this.setLocation(200, 100);
        this.createGUI();
    }

    /**
     * Used to create a JavaFX panel
     */
    private void createGUI()
    {
        panel = new JFXPanel();

        Platform.runLater(new Runnable()
        {
            @Override
            public void run()
            {
                panel.setScene(createScene());
                SwingUtilities.invokeLater(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        packGUI();
                    }
                });
            }
        });
    }

    /**
     * Creates a JavaFX scene
     *
     * @return the JavaFX scene
     */
    private Scene createScene()
    {
        Parent root = null;
        try
        {
            FXMLLoader loader = new FXMLLoader(getClass().getResource("FXMLDescripti
            FXMLDescriptionDocumentController controller = new FXMLDescriptionDocume
            loader.setController(controller);
            loader.setClassLoader(getClass().getClassLoader());
            root = (Parent)loader.load();
            System.out.println(loader.getController().toString());
            System.out.println(loader.getLocation());
            return new Scene(root);
        } catch (Exception e)
        {
            e.printStackTrace();
        }
        return new Scene(root);
```

```
    }

    /**
     * Packs the GUI and sets it visible
     */
    public void packGUI()
    {
        this.add(panel);
        this.setVisible(true);
    }

    public void windowActivated(WindowEvent event)
    {
    }

    public void windowClosed(WindowEvent event) {
        StateManager.setDescriptionGUI(null);
    }

    public void windowClosing(WindowEvent event)
    {
    }

    public void windowDeactivated(WindowEvent event)
    {
    }

    public void windowOpened(WindowEvent event)
    {
    }

    public void windowDeiconified(WindowEvent event)
    {
    }

    public void windowIconified(WindowEvent event)
    {
    }
}
```

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

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

---

```
package Extension.GUI;

import javafx.application.Platform;
import javafx.embed.swing.JFXPanel;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;

import Extension.BackEnd.*;
```

```
/**
 * The Graphical User Interface used to show the user the list of exercises along wi
 * and allows them to load an exercise
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class ExerciseListGUI extends JFrame
{
    private JFXPanel panel;

    /**
     * Creates the GUI for the extension
     */
    public ExerciseListGUI()
    {
        this.setTitle("BlueJ TA - Exercise List");
        this.setDefaultCloseOperation(HIDE_ON_CLOSE);
        this.setSize(605, 530);
        this.setResizable(false);
        this.setLocationRelativeTo(null);
        this.setLocation(100, 100);
        this.createGUI();
    }

    /**
     * Used to create a JavaFX panel
     */
    private void createGUI()
    {
        panel = new JFXPanel();

        Platform.runLater(new Runnable()
        {
            @Override
            public void run()
            {
                panel.setScene(createScene());
                SwingUtilities.invokeLater(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        packGUI();
                    }
                });
            }
        });
    }

    /**
     * Creates a JavaFX scene
```

```
     *
     * @return the JavaFX scene
     */
    private Scene createScene()
    {
        Parent root = null;
        try
        {
            FXMLLoader loader = new FXMLLoader(getClass().getResource("FXMLDocument.
            FXMLExerciseDocumentController controller = new FXMLExerciseDocumentCont
            loader.setController(controller);
            loader.setClassLoader(getClass().getClassLoader());
            root =  (Parent)loader.load();
            System.out.println(loader.getController().toString());
            System.out.println(loader.getLocation());
            return new Scene(root);
        } catch (Exception e)
        {
            e.printStackTrace();
        }
        return new Scene(root);
    }

    /**
     * Packs the GUI and sets it visible
     */
    public void packGUI()
    {
        this.add(panel);
        this.setVisible(true);
    }
}
```

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

BEGIN File:
/Projects/BlueJ_TA/Code/src/Extension/GUI/FXMLDescriptionDocumentController.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;
import javafx.beans.value.ObservableValue;
import javafx.stage.Window;

import Extension.BackEnd.*;

/**
 * Used by the FXML document to control the description GUI
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class FXMLDescriptionDocumentController implements Initializable
{

    @FXML
    private Stage stage;
    @FXML
    private ListView<Exercise> list;
    @FXML
    private Label titleLabel;
    @FXML
    private Label descriptionLabel;
    @FXML
    private Label examplesLabel;
    @FXML
    private Button okButton;
    private Exercise exercise;

     /**
     * Called whenever a button or menu item is pressed, and calls a function
     * depending on the button/item pressed.
     *
     * @param event the calling event
     */
    @FXML
    private void handleButtonAction(ActionEvent event)
    {
        if(event.getSource().equals(okButton))
        {
            StateManager.disposeDescriptionGUI();
        }
    }

    /**
     * Initializes this GUI with the proper data
     *
     * @param url   not used
     * @param rb    not used
     */
    public void initialize(URL url, ResourceBundle rb)
```

```
    {
        if(StateManager.getCurrentExercise() != null)
        {
            exercise = StateManager.getCurrentExercise();
            titleLabel.setText(exercise.getTitle());
            descriptionLabel.setText(exercise.getDescription());
            String examples = "";
            List<String> examp = exercise.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/FXMLDescriptionDocumentController.java

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;
import javafx.beans.value.ObservableValue;
import javafx.stage.Window;

import Extension.BackEnd.*;

/**
 * Used by the FXML document to control the ExerciseList GUI.
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
```

```
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class FXMLExerciseDocumentController implements Initializable
{
    @FXML
    private Stage stage;
    @FXML
    private ListView<Exercise> list;
    @FXML
    private Label titleLabel;
    @FXML
    private Label descriptionLabel;
    @FXML
    private Label examplesLabel;
    @FXML
    private Button submitButton;

    private Exercise exerciseSelected;

    /**
     * Called whenever a button or menu item is pressed, and calls a function
     * depending on the button/item pressed.
     *
     * @param event the calling event
     */
    @FXML
    private void handleButtonAction(ActionEvent event)
    {
        if(event.getSource().equals(submitButton))
        {
            StateManager.setCurrentExercise(exerciseSelected);
            StateManager.getExerciseListGUI().setVisible(false);
            StateManager.disposeDescriptionGUI();
            StateManager.disposeTestResultsGUI();
            new Thread()
            {
                public void run()
                {
                    StateManager.getCurrentExercise().launch();
                }
            }.start();
        }
    }

    /**
     * Used to initialize the GUI
     *
     * @param url unused
     * @param rb unused
     */
    @SuppressWarnings("restriction")
    @Override
    public void initialize(URL url, ResourceBundle rb)
    {
        ObservableList<Exercise> items = FXCollections.observableArrayList();
        items.setAll(StateManager.getExerciseList());
```

```
        list.setItems(items);
        list.getSelectionModel().selectedItemProperty().addListener(
            new ChangeListener<Exercise>()
            {
                public void changed(ObservableValue<? extends Exercise> ov, Exercise
                                    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

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

---

```
package Extension.GUI;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.Initializable;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.Label;
import org.junit.runner.Result;
import javax.swing.JOptionPane;
import javafx.scene.shape.Circle;
import java.io.File;

import Extension.BackEnd.*;

import javafx.application.Platform;

/**
 * Used by the FXML document to control the test results GUI
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
```

```
 * @version April2015
 */
public class FXMLTestResultsDocumentController implements Initializable
{
    @FXML
    private Button runButton;
    @FXML
    private TextArea textArea;
    @FXML
    private Label titleLabel;

    /**
     * Called whenever a button or menu item is pressed, and calls a function
     * depending on the button/item pressed.
     *
     * @param event the calling event
     */
    @FXML
    private void handleButtonAction(ActionEvent event)
    {
        if(event.getSource().equals(runButton))
        {
            textArea.clear();
            runTest();
        }
    }

    /**
     * Gets the results from running the JUnit tests
     * then formats the failures and prints it to the text area.
     */
    public void runTest()
    {
        if( StateManager.getCurrentExercise() != null)
        {
            new Thread()
            {
                public void run()
                {
                    final String resultText = StateManager.getCurrentExercise().exec
                    Platform.runLater(new Runnable()
                    {
                        public void run()
                        {
                            textArea.setText(resultText);
                        }
                    });
                }
            }.start();
        }else
        {
            textArea.setText("Exercise not loaded");
        }
    }

    /**
     * Used to initialize this GUI
     */
```

```
    @Override
    public void initialize(URL url, ResourceBundle rb)
    {
        textArea.setEditable(false);
        if(StateManager.getCurrentExercise() != null)
        {
            titleLabel.setText(StateManager.getCurrentExercise().getTitle());
            runTest();
        }else
        {
            textArea.setText("Exercise not loaded");
        }
    }
}
```

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

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

---

```
package Extension.GUI;

import javafx.application.Platform;
import javafx.embed.swing.JFXPanel;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import java.awt.event.WindowListener;
import java.awt.event.WindowEvent;

import static javax.swing.JFrame.EXIT_ON_CLOSE;

import Extension.BackEnd.*;

/**
 * The Graphical User Interface used to display test result data to the user after t
 * have been run on the user's code.
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class TestResultsGUI extends JFrame implements WindowListener
{
    private JFXPanel panel;

    /**
     * Creates the test results GUI
     */
```

```
public TestResultsGUI()
{
    this.setTitle("BlueJ TA - Test Results");
    this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    this.setSize(457, 471);
    this.addWindowListener(this);
    this.setResizable(false);
    this.setLocationRelativeTo(null);
    this.setLocation(200, 100);
    this.createGUI();
}

/**
 * Used to create a JavaFX panel
 */
private void createGUI()
{
    panel = new JFXPanel();

    Platform.runLater(new Runnable()
    {
        @Override
        public void run()
        {
            panel.setScene(createScene());
            SwingUtilities.invokeLater(new Runnable()
            {
                @Override
                public void run()
                {
                    packGUI();
                }
            });
        }
    });
}

/**
 * Creates a JavaFX scene
 *
 * @return the JavaFX scene
 */
private Scene createScene()
{
    Parent root = null;
    try {
        FXMLLoader loader = new FXMLLoader(getClass().getResource("FXMLTestDocum
        FXMLTestResultsDocumentController controller = new FXMLTestResultsDocume
        loader.setController(controller);
        loader.setClassLoader(getClass().getClassLoader());
        root =  (Parent)loader.load();
        System.out.println(loader.getController().toString());
        System.out.println(loader.getLocation());
        return new Scene(root);
    } catch (Exception e)
    {
        e.printStackTrace();
    }
```

```
        return new Scene(root);
    }

    /**
     * Packs the GUI and sets it visible
     */
    public void packGUI()
    {
        this.add(panel);
        this.setVisible(true);
    }

    public void windowActivated(WindowEvent event)
    {
    }

    public void windowClosed(WindowEvent event)
    {
        StateManager.setTestResultsGUI(null);
    }

    public void windowClosing(WindowEvent event)
    {
        StateManager.setTestResultsGUI(null);
    }

    public void windowDeactivated(WindowEvent event)
    {
    }

    public void windowOpened(WindowEvent event)
    {
    }

    public void windowDeiconified(WindowEvent event)
    {
    }

    public void windowIconified(WindowEvent event)
    {
    }

}
```

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

BEGIN File: /Projects/BlueJ_TA/Code/src/Extension/Main.java

```
package Extension;

import bluej.extensions.BlueJ;
import bluej.extensions.Extension;

import Extension.BackEnd.*;
```

```
/**
 * The base point of this BlueJ extension.
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class Main extends Extension
{

        /**
     * Used to get the name of this extension
     *
     * @return the name of this extension
     */
    @Override
        public String getName()
        {
                return "BlueJ TA";
        }

        /**
     * Used to get the current version of this extension
     *
     * @return the current version of this extension
     */
    @Override
        public String getVersion()
        {
                return "April2015";
        }

    /**
     * Used to check if this extension is compatible with the BlueJ
     * version used
     *
     * @return true if this extension is compatible, false otherwise
     */
    @Override
        public boolean isCompatible()
        {
                return true;
        }

        /**
         * Called when this extension is being terminated
         */
        @Override
        public void terminate()
        {
            System.out.println("BlueJ TA Terminated");
        }

        /**
```

```
     * Used at start up of the extension
     *
     * @param blueJ The BlueJ instance
     */
    @Override
        public void startup(BlueJ blueJ)
        {
                blueJ.setMenuGenerator(new MenuBuilder());
        blueJ.setPreferenceGenerator(new Preferences(blueJ));
                StateManager.setup(blueJ);
        }
}
```

END File: /Projects/BlueJ_TA/Code/src/Extension/Main.java

BEGIN File: /Projects/BlueJ_TA/Code/src/Extension/MenuBuilder.java

---

```
package Extension;

import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.AbstractAction;
import java.awt.event.ActionEvent;
import bluej.extensions.BPackage;
import bluej.extensions.BClass;
import bluej.extensions.MenuGenerator;

import Extension.BackEnd.*;

/**
 * Creates menu items for this BlueJ extension
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class MenuBuilder extends MenuGenerator
{
    /**
     * Adds extension menu items to the BlueJ tools menu
     *
     * @param bPackage  BlueJ package
     * @return     the menu item to be added to BlueJ tools menu
     */
    @Override
    public JMenuItem getToolsMenuItem(BPackage bPackage)
    {
        return new JMenuItem(new AbstractAction("BlueJ TA")
        {
            @Override
            public void actionPerformed(ActionEvent event)
```

```
            {
                StateManager.createExerciseListGUI();
            }
        });
    }

    /**
     * Adds extension menu items to the BlueJ class card
     *
     * @Param bClass    the selected Class
     * @return  the menu item to be added to the Class card of the selected Class
     */
    @Override
    public JMenuItem getClassMenuItem(BClass bClass)
    {
        if(StateManager.getCurrentExercise() == null ||
            !StateManager.getCurrentExercise().getJavaName().equals(bClass.getName()))
        {
            return null;
        }

        JMenu jm = new JMenu("BlueJ TA");
        jm.add(new JMenuItem(new AbstractAction("Exercise Description")
        {
            @Override
            public void actionPerformed(ActionEvent event)
            {
                if(StateManager.getCurrentExercise() == null)
                {
                    StateManager.showErrorMessage("No Exercise Selected\n" +
                    "Go to Tools->CodingBat to pick exercise");
                }else
                {
                    StateManager.createDescriptionGUI();
                }
            }
        }));

        jm.add(new JMenuItem(new AbstractAction("Execute")
        {
            @Override
            public void actionPerformed(ActionEvent event)
            {
                if(StateManager.getCurrentExercise() == null)
                {
                    StateManager.showErrorMessage("No Exercise Selected\n" +
                    "Go to Tools->CodingBat to pick exercise");
                }else
                {
                    StateManager.createTestResultsGUI();
                }
            }
        }));
        return jm;
    }
}
```

END File: /Projects/BlueJ_TA/Code/src/Extension/MenuBuilder.java

BEGIN File: /Projects/BlueJ_TA/Code/src/Extension/Preferences.java

---

```
package Extension;

import bluej.extensions.*;
import bluej.extensions.event.*;
import bluej.extensions.editor.*;

import java.net.URL;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.GroupLayout.*;
import java.awt.FlowLayout;
import java.io.File;

import Extension.BackEnd.*;

/**
 * This class is used to generate a Panel within BlueJ Preferences->Extension
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class Preferences implements PreferenceGenerator
{
    private JPanel myPanel, localPanel, uriPanel;
    private JLabel localDir, uriDir;
    private JTextField uriTextfield, localTextfield;
    private JButton localButton, uriButton;
    private BlueJ bluej;
    public static final String PROFILE_LABEL="Exercise-Location-BlueJTA";

    /**
     * Creates the Panel within BlueJ's Preferences
     *
     * @param bluej the BlueJ instance
     */
    public Preferences(BlueJ bluej)
    {
        this.bluej = bluej;
        myPanel = new JPanel();
        localPanel = new JPanel();
        uriPanel = new JPanel();

        localDir = new JLabel("Local Exercise Directory");
        uriDir = new JLabel("URI Exercise Location");
        uriTextfield = new JTextField (40);
        localTextfield =new JTextField (40);

        localButton = new JButton("Load");
        localButton.addActionListener(new ActionListener()
        {
```

```
            @Override
            public void actionPerformed(ActionEvent e)
            {
                String location = openFileChooser();
                localTextfield.setText(location);

                StateManager.setExercisesLocation(location);
                StateManager.parseExercises();
                StateManager.setExerciseListGUI(null);
            }
        });

        uriButton = new JButton("Connect");
        uriButton.addActionListener(new ActionListener()
        {
           @Override
           public void actionPerformed(ActionEvent e)
           {
               //to be implemented
           }
        });

        BoxLayout layout = new BoxLayout(myPanel, BoxLayout.Y_AXIS);
        myPanel.setLayout(layout);

        localPanel.add (new JLabel ("Local Exercise Directory"));
        localPanel.add(localTextfield);
        localPanel.add(localButton);

        myPanel.add(localPanel);

        uriPanel.add (new JLabel ("URI Exercise Location"));
        uriPanel.add (uriTextfield);
        uriPanel.add(uriButton);

        myPanel.add(uriPanel);


        uriTextfield.setText("Currently not implemented");
        uriTextfield.setEditable(false);

        loadValues();
    }

    /**
     * Opens a JFileChooser for the user to pick the location of the exercises
     *
     * @return  the path of the exercises specified by the user.
     */
    private String openFileChooser()
    {
        JFileChooser fc = new JFileChooser();
        fc.setCurrentDirectory(new File("."));
        fc.setDialogTitle("Choose Exercise Location");
        fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        fc.setAcceptAllFileFilterUsed(false);

        if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION)
```

```
        {
            return fc.getSelectedFile().getAbsolutePath();
        } else
        {
            return "";
        }
    }

    /**
     * Returns the JPanel created by this object
     *
     * @return  the populated JPanel
     */
    @Override
    public JPanel getPanel ()
    {
        return myPanel;
    }

    /**
     * Used to save the path given by the user into the BlueJ properties file
     */
    @Override
    public void saveValues ()
    {
        bluej.setExtensionPropertyString(PROFILE_LABEL, localTextfield.getText());
    }

    /**
     * Used to load the previously saved path given by the user from the
     * BlueJ properties file
     */
    @Override
    public void loadValues ()
    {
        String exLocation = bluej.getExtensionPropertyString(PROFILE_LABEL, "");
        localTextfield.setText(exLocation);
        if(exLocation != null && !exLocation.isEmpty())
        {
            StateManager.setExercisesLocation(exLocation);
            StateManager.parseExercises();
        }
    }
}
```

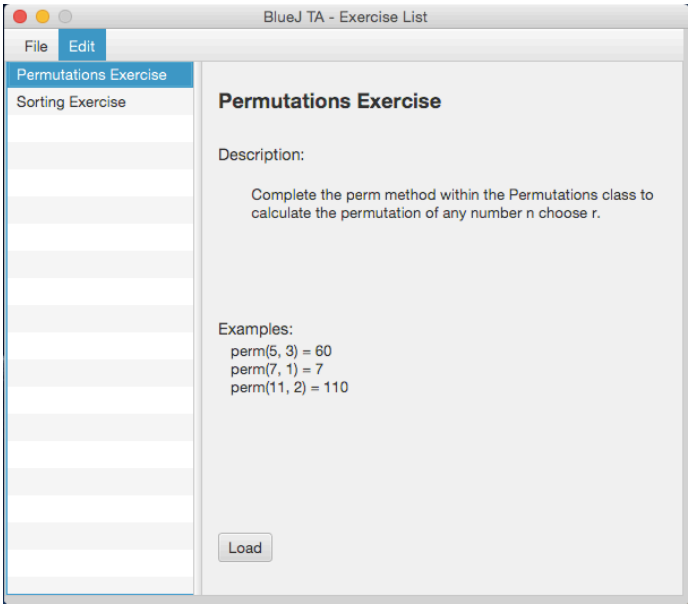
END File: /Projects/BlueJ_TA/Code/src/Extension/Preferences.java

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 objec
 * 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
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class XMLReader
{
    private JAXBContext context;
    private Unmarshaller um;
    private Exercise ex;

    /**
     * Constructor
     */
    public XMLReader()
    {
    }

    /**
     * Reads an XML file that has been converted into an InputStream
     *
     * @param is    the XML file as InputStream
     * @return  the Exercise object created from the XML file.
     */
    public Exercise readExercise(InputStream is)
    {
        ex = null;
        try
```

```
        {
            context=JAXBContext.newInstance(Exercise.class);
            um=context.createUnmarshaller();
            ex=(Exercise) um.unmarshal(new InputStreamReader( is ) );
        } catch (JAXBException ex)
        {
            ex.printStackTrace();
        }
        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
 */
public class JRunner
{

    /**
     * A flag used to signal when its ok for the extension to remove
     * this class and the JUnit test class files from the exercise project.
     */
    public static boolean running = false;

    /**
     * Resets the static running flag for a future test to be run.
     */
    public static void resetRunning()
    {
        running = false;
    }

    /**
     * Executes the JUnit Test of a given JUnit class and returns the results in
     * the form of a string.
     *
     * @param testClass the name of the JUnit test class to execute.
     * @return the results of the test
     */
    public static String run(String testClass)
    {
```

```
        Class testC = null;
        try
        {
            testC = Class.forName(testClass);
        } catch (Exception e)
        {
            return e.getMessage();
        }

        //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
-->
<project name = "BlueJ-TA" default = "run" basedir = ".">
    <property file = "project.properties"/>
    <property name = "dist.dir" location = "dist"/>
    <property name = "source.dir" location = "src"/>
    <property name = "compile.dest.dir" location = "${source.dir}"/>
    <property name = "lib.dir" location = "${source.dir}/+libs"/>
    <property name = "doc.dir" location = "doc"/>
    <property name = "results.dir" location = "results"/>

    <!-- Locates JavaFX. -->
    <pathconvert property="javafx.jar.file" setonempty="false" pathsep=" ">
        <path>
            <fileset dir="${java.home}/lib/" includes="**/jfxrt.jar" />
        </path>
    </pathconvert>

    <!-- Set project class path. -->
    <path id="cp">
        <pathelement path = "${compile.dest.dir}"/>
        <file file = "${bluej.lib.dir}/bluejext.jar"/>
        <fileset dir = "${lib.dir}" includes = "*.jar"/>
        <file file = "${javafx.jar.file}"/>
        <file file = "${junit.jar.file}"/>
    </path>
```

```
    <!-- Detects JavaFX jar. -->
    <condition property="javafx.jar.exists">
        <resourcecount when="greater" count="0">
            <fileset file="${javafx.jar.file}"/>
        </resourcecount>
    </condition>

    <!-- Detect Bad Java versions -->
    <condition property = "bad.java.version">
        <or>
            <equals arg1="${ant.java.version}" arg2="1.1"/>
            <equals arg1="${ant.java.version}" arg2="1.2"/>
            <equals arg1="${ant.java.version}" arg2="1.3"/>
            <equals arg1="${ant.java.version}" arg2="1.4"/>
            <equals arg1="${ant.java.version}" arg2="1.5"/>
            <equals arg1="${ant.java.version}" arg2="1.6"/>
        </or>
    </condition>

    <!-- Set error messages. -->
    <property name = "error.message.bad.java.version" value = "Unsupported Java
version: ${ant.java.version}.
            Make sure that the Java version is 1.7 or greater."/>
    <property name = "error.message.junit.not.found" value = "The build could
not locate JUnit."/>
    <property name = "error.message.javafx.not.found" value = "The build could
not locate JavaFX."/>
    <property name = "error.message.bluej.lib.dir.not.defined"
     value = "bluej.lib.dir property does not have a value."/>
    <property name = "error.message.bluej.lib.dir.does.not.exit"
     value = "bluej.lib.dir does not refer to real folder."/>
    <property name = "error.message.user.bluej.dir.does.not.exit" value =
"user.bluej.dir does not refer to real folder."/>

    <!-- Detects Mac. -->
    <condition property="isMac">
        <os family="mac" />
    </condition>

    <!-- Detects Windows. -->
    <condition property="isWindows">
        <os family="windows" />
    </condition>

    <!-- Detects Unix. -->
    <condition property="isLinuxNotMac">
        <and>
          <os family="unix" />
          <not>
              <os family="mac" />
          </not>
        </and>
    </condition>

    <!-- Loads Windows properties. -->
    <target name = "windows-props" if = "isWindows">
        <property file = "local.properties.windows"/>
    </target>
```

```
<!-- Loads Mac properties. -->
<target name = "mac-props" if = "isMac">
    <property file = "local.properties.macOSX"/>
</target>

<!-- Loads Unix properties. -->
<target name = "linux-props" if = "isLinuxNotMac">
    <property file = "local.properties.unix"/>
</target>

<!-- Sets properties that depend on OS specific properties. -->
<target name = "set-env" depends = "windows-props,mac-props,linux-props">
    <property name = "user.bluej.dir" location = "${user.home}/.bluej"/>
    <property name = "ext.dir" location = "${user.bluej.dir}/extensions"/>
    <property name = "jar.file" location = "${dist.dir}/${jar.name}"/>
    <property name = "ext.file" location = "${ext.dir}/${jar.name}"/>
    <property name = "bluej.log.file"
              location = "${user.bluej.dir}/bluej-debuglog.txt"/>

    <!-- Locates JUnit. -->
    <pathconvert property="junit.jar.file" setonempty="false" pathsep=" ">
        <path>
            <fileset dir="${bluej.lib.dir}" includes="**/junit*.jar" />
        </path>
    </pathconvert>

    <!-- Detects JUnit jar. -->
    <condition property="junit.jar.exists">
        <available file = "${junit.jar.file}"/>
    </condition>

    <!-- Make sure the BlueJ lib dir exists -->
    <condition property="bluej.lib.dir.exists">
        <available file = "${bluej.lib.dir}"/>
    </condition>

</target>

<!-- Delete build-generated files. -->
<target name = "clean" depends = "set-env"
        description = "Removes build-generated files.">
    <delete file = "${ext.file}"/>
    <delete dir = "${dist.dir}"/>
    <delete>
        <fileset dir="${source.dir}" includes="**/*.class"/>
        <fileset dir="${source.dir}" includes="**/*.ctxt"/>
    </delete>
    <delete dir = "${doc.dir}"/>
    <delete dir = "${results.dir}"/>
</target>

<!-- Create folders needed for the build. -->
<target name = "prebuild"
        description = "Creates build dependencies.">
    <mkdir dir = "${dist.dir}"/>
    <mkdir dir = "${compile.dest.dir}"/>
</target>
```

```
<!-- Compile sources. -->
<target name = "compile" depends = "set-env,prebuild"
        description = "Compiles Java sources.">
    <fail message = "${error.message.bluej.lib.dir.not.defined}"
    unless = "bluej.lib.dir"/>
    <fail message = "error.message.bluej.lib.dir.does.not.exit"
     unless = "bluej.lib.dir.exists"/>
    <fail message = "${error.message.javafx.not.found}"
     unless = "javafx.jar.exists"/>
    <fail message = "${error.message.junit.not.found}"
     unless = "junit.jar.exists"/>
    <fail message="${error.message.bad.java.version}"
     if = "bad.java.version"/>
    <javac srcdir = "${source.dir}" destdir = "${compile.dest.dir}"
     debug = "${debug.mode}" source = "${source.level}"
     target = "${target.level}" includeantruntime="false"
     bootclasspath="${java.home}/lib/rt.jar"
     excludes = "Extension/BackEnd/runner/JRunner.java">
        <classpath refid = "cp"/>
        <!--<compilerarg value="-Xlint"/>-->
    </javac>

    <!--The JRunner needs to be compiled so class ends up
       in the same folder stead of the src/ folder.-->
    <javac srcdir = "${source.dir}/Extension/BackEnd/runner/"
     destdir = "${compile.dest.dir}/Extension/BackEnd/runner/"
     debug = "${debug.mode}" source = "${source.level}"
     target = "${target.level}" includeantruntime="false"
     bootclasspath="${java.home}/lib/rt.jar">
        <classpath refid = "cp"/>
        <!--<compilerarg value="-Xlint"/>-->
    </javac>
</target>

<!-- Builds the jar file and includes JGit classes.-->
<target name = "jar" depends = "compile"
        description = "Creates the extension Java jar file.">
    <jar destfile = "${jar.file}">
        <fileset dir = "${compile.dest.dir}" includes="**/*.class"/>
        <fileset dir = "${source.dir}" includes="**/*.fxml"/>
        <fileset dir = "${source.dir}" includes="**/*.lst"/>
        <fileset dir = "${source.dir}" includes="**/*.xml"/>
        <zipfileset excludes="META-INF/**/*" src="${javafx.jar.file}"/>
        <manifest>
            <attribute name = "Main-Class" value = "${main.class}"/>
            <attribute name = "Class-Path" value = "${jar.class.path}"/>
        </manifest>
    </jar>
</target>

<!--
 Copies the extension and it's suppporting JARs into the user's folder.-->
<target name = "install-ext" depends = "jar"
 description = "Puts the Java jar into the user's extension folder.">
    <mkdir dir = "${ext.dir}"/>
    <copy file = "${jar.file}" todir="${ext.dir}"/>
</target>
```

```
<!-- Performs launch-required steps. -->
<target name = "pre-launch" depends = "install-ext">
    <fail message="${error.message.bad.java.version}"
     if = "bad.java.version"/>
    <delete file = "${bluej.log.file}"/>
    <mkdir dir = "${results.dir}"/>
</target>

<!-- Runs the program on non-Unix platforms. -->
<target name = "launch-1" depends = "pre-launch"
 unless = "isLinuxNotMac">
    <exec executable="${bluej.launch.command}"/>
</target>

<!-- Runs the program on a Unix machine. -->
<target name = "launch-alt" depends = "pre-launch"
 if = "isLinuxNotMac">
    <java
     fork = "true"
     classname = "bluej.Boot"
     failonerror = "true" >
        <arg value = "-Dawt.useSystemAAFontSettings=on"/>
        <classpath>
            <pathelement location="${java.home}/lib/tools.jar"/>
            <pathelement location="${bluej.lib.dir}/bluej.jar"/>
        </classpath>
    </java>
</target>

<!-- Runs the program. -->
<target name = "launch" depends = "launch-1,launch-alt"
 description = "Launches BlueJ"/>

<!-- Prints and saves BlueJ's output. -->
<target name = "capture-log" depends = "launch"
 description = "Prints BlueJ's logs then saves them to the ${results.dir}">
    <!-- Make sure the BlueJ lib dir exists -->
    <condition property="user.bluej.dir.exists">
        <available file = "${user.bluej.dir}"/>
    </condition>
    <fail message = "${error.message.user.bluej.dir.does.not.exit}"
     unless = "user.bluej.dir.exists"/>
    <echo message = "------------------------------------------------"/>
    <echo message = "BEGIN BlueJ's output:"/>
    <echo message = "------------------------------------------------"/>
    <concat>
        <fileset file = "${bluej.log.file}"/>
    </concat>
    <echo message = "------------------------------------------------"/>
    <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);
            }
            return 1;
        }
    </sampleAnswer>
    <javaName>Permutations</javaName>
    <javaCode>public class Permutations {
    public int perm(int n, int r){
        return -1;
    }
}
    </javaCode>
    <junitName>PermutationTest</junitName>
    <junitCode>
        import org.junit.*;
        import static org.junit.Assert.*;

        public class PermutationTest {

            static Permutations userPerm = new Permutations();

            @Test
            public void permutationTestA(){
                assertPerm(5, 3);
            }

            @Test
            public void permutationTestB(){
```

```
            assertPerm(10, 2);
        }

        @Test
        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++){
                    for(int k = i+1; k &lt; data.length; k++){
                        if(data[i] &gt; data[k]){
                            swap(i, k, data);
                        }
```

```
                    }
                }
            }

            private static void swap(int a, int b, int[] data){
                int temp = data[a];
                data[a] = data[b];
                data[b] = temp;
            }
        }
    </sampleAnswer>
    <javaName>SortingExercise</javaName>
    <javaCode>public class SortingExercise {
    public void sort(int[] data){
        //add your code here
    }
}
    </javaCode>
    <junitName>SortTest</junitName>
    <junitCode>
        import org.junit.*;
        import static org.junit.Assert.*;
        import java.util.Random;

        public class SortTest {

            static SortingExercise sortEx = new SortingExercise();
            static Random rand = new Random();

            @Test
            public void testSort(){
                int[] data = randData(1000, -100, 100);
                sortEx.sort(data);
                assertSorted(data, true);
            }

            private static int[] randData(int size, int min, int max){
                int[] data = new int[size];
                for(int i = 0; i &lt; data.length; i++){
                    data[i] = rand.nextInt((max - min) + 1) + min;
                }
                return data;
            }

            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

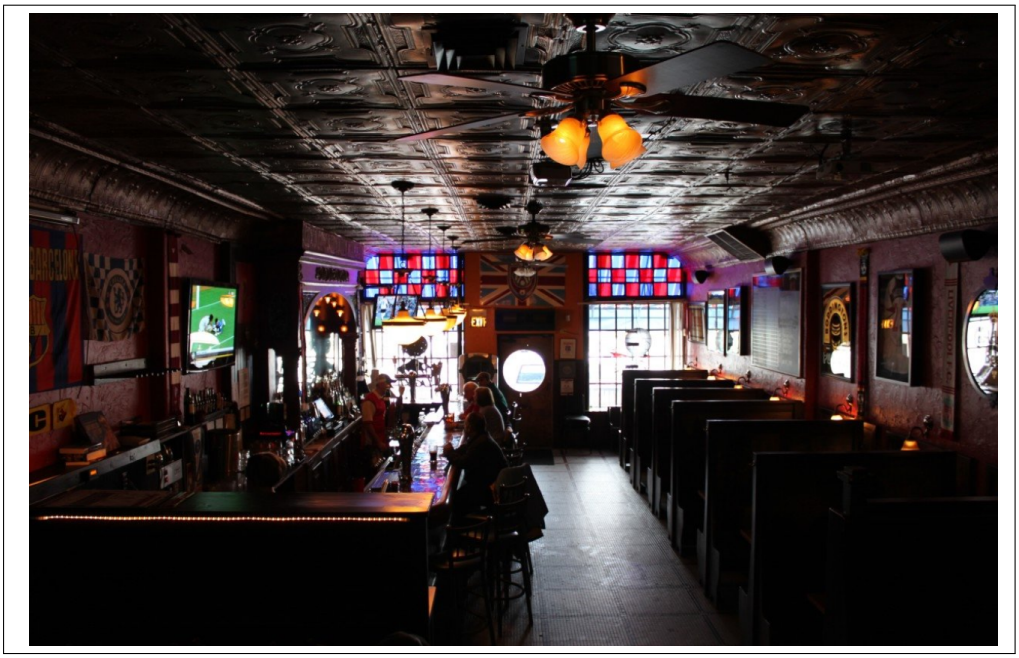
Note: 1 artifacts were ommitted in this abridged version.

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.

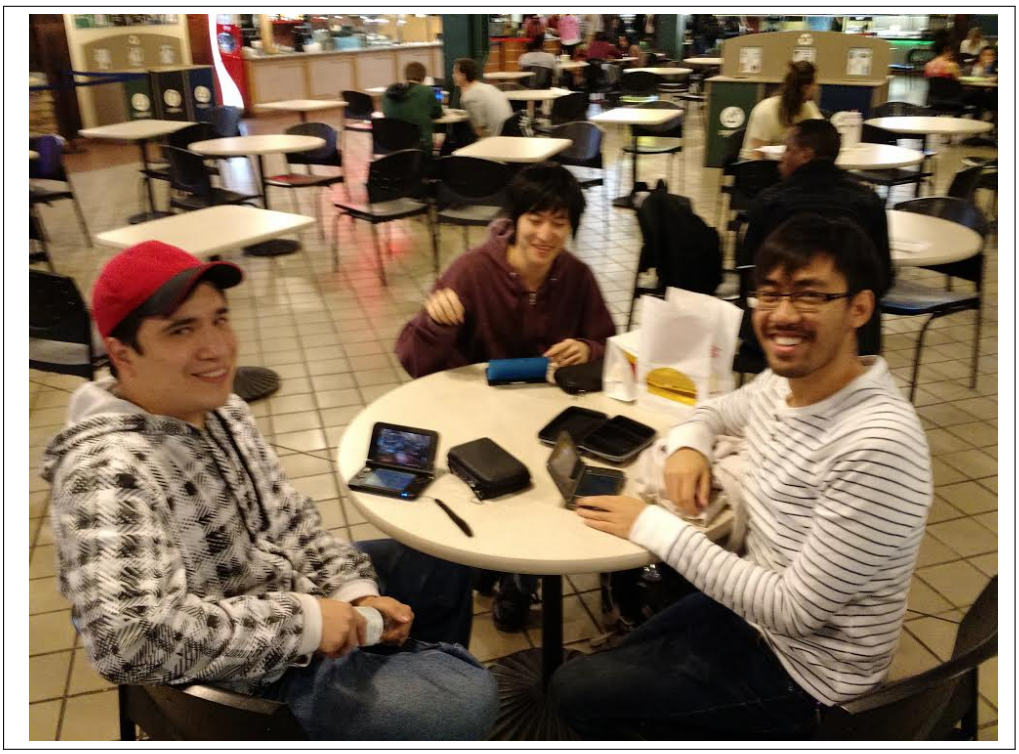


Team Building @ Euclid Hall

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.

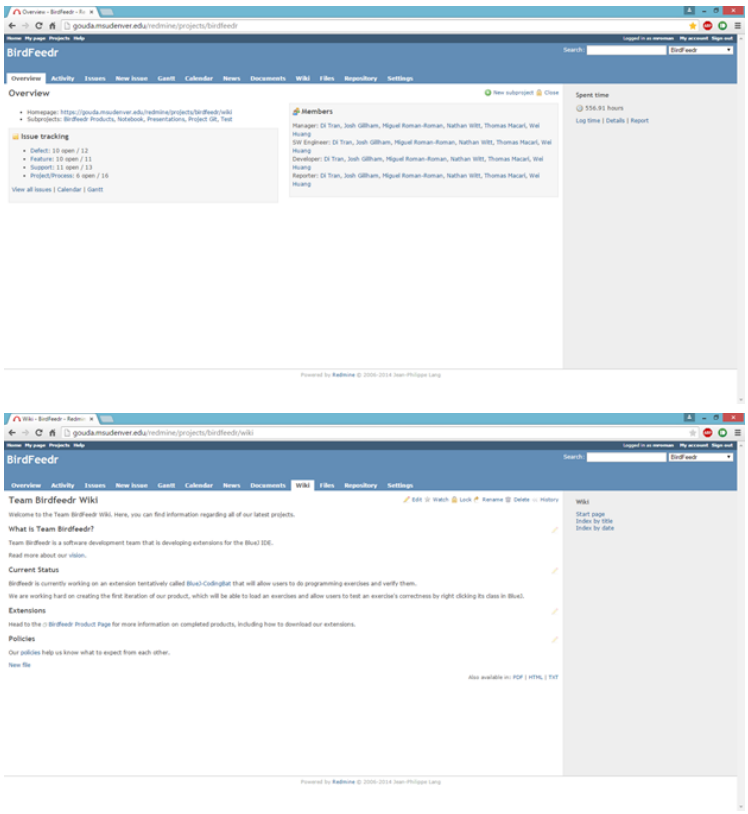


Tools

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.

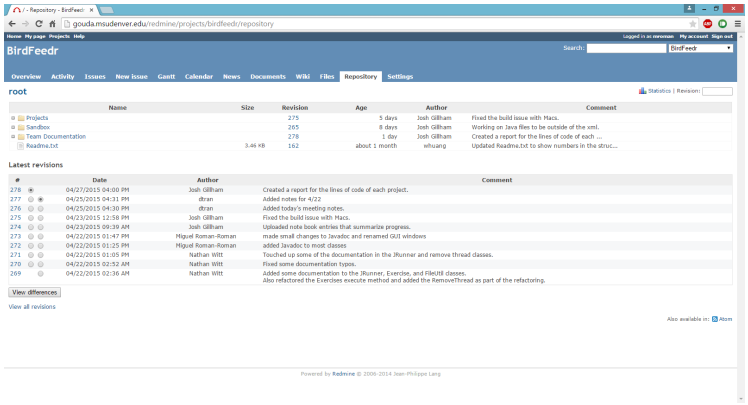
Redmine

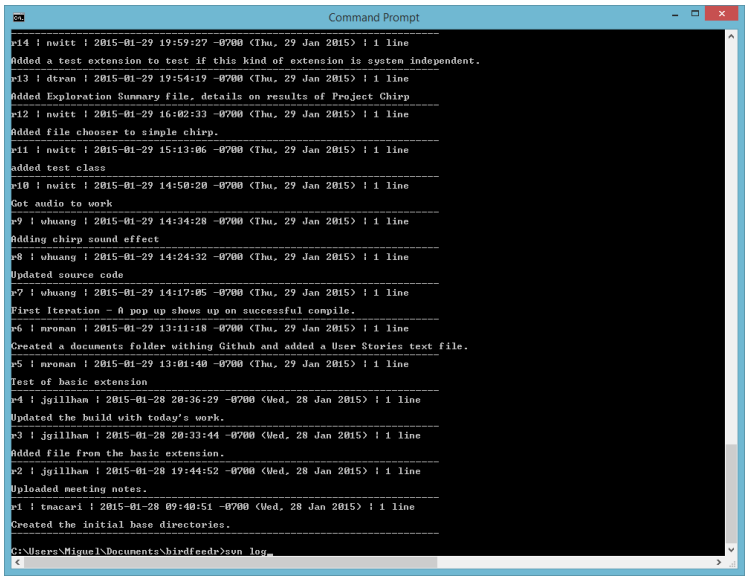
Description Redmine is the web based project management tool used by our team. Team hours, project issues, and wiki page were placed here. This tool gave stakeholders an access point to what we were doing. The following artifact consists of two images of BirdFeedr's Redmine.



Subversion SVN

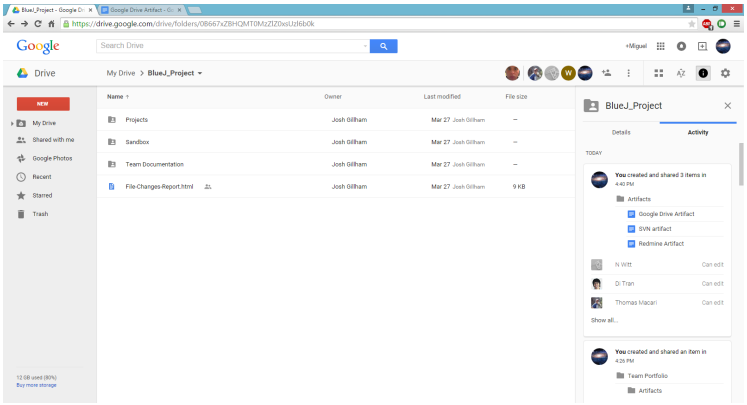
Description Subversion is Team BirdFeedrs version control system, and this repository contains all of our produced work. Since the repository is version controlled, the team was also able to revert back to previous versions of any files if necessary. This tool allowed team members to share the product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





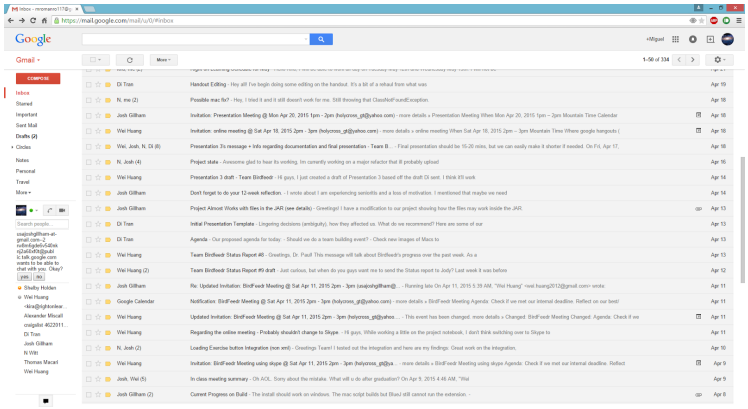
Google Drive

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.



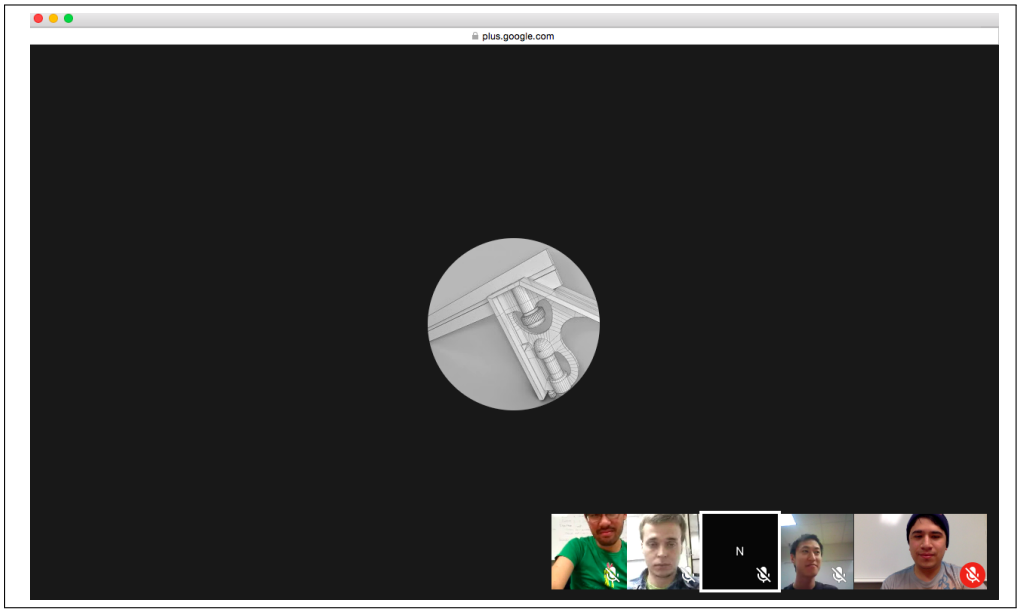
Google Hangouts

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;
import javafx.beans.value.ObservableValue;
import javafx.stage.Window;

import Extension.BackEnd.*;

/**
 * Used by the FXML document to control the ExerciseList GUI.
 *
 * @author Miguel Roman-Roman
 * @author Wei Huang
 * @author Di Tran
 * @author Josh Gillham
 * @author Nathan Witt
 * @author Thomas Macari
 * @version April2015
 */
public class FXMLExerciseDocumentController implements Initializable
{
    @FXML
    private Stage stage;
    @FXML
    private ListView<Exercise> list;
    @FXML
    private Label titleLabel;
    @FXML
```

```
private Label descriptionLabel;
@FXML
private Label examplesLabel;
@FXML
private Button submitButton;

private Exercise exerciseSelected;

/**
 * Called whenever a button or menu item is pressed, and calls a function
 * depending on the button/item pressed.
 *
 * @param event the calling event
 */
@FXML
private void handleButtonAction(ActionEvent event)
{
    if(event.getSource().equals(submitButton))
    {
        StateManager.setCurrentExercise(exerciseSelected);
        StateManager.getExerciseListGUI().setVisible(false);
        StateManager.disposeDescriptionGUI();
        StateManager.disposeTestResultsGUI();
        new Thread()
        {
            public void run()
            {
                StateManager.getCurrentExercise().launch();
            }
        }.start();
    }
}

/**
 * Used to initialize the GUI
 *
 * @param url unused
 * @param rb unused
 */
@SuppressWarnings("restriction")
@Override
public void initialize(URL url, ResourceBundle rb)
{
    ObservableList<Exercise> items = FXCollections.observableArrayList();
    items.setAll(StateManager.getExerciseList());
    list.setItems(items);
    list.getSelectionModel().selectedItemProperty().addListener(
        new ChangeListener<Exercise>()
        {
            public void changed(ObservableValue<? extends Exercise> ov, Exercise
                                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++)
```

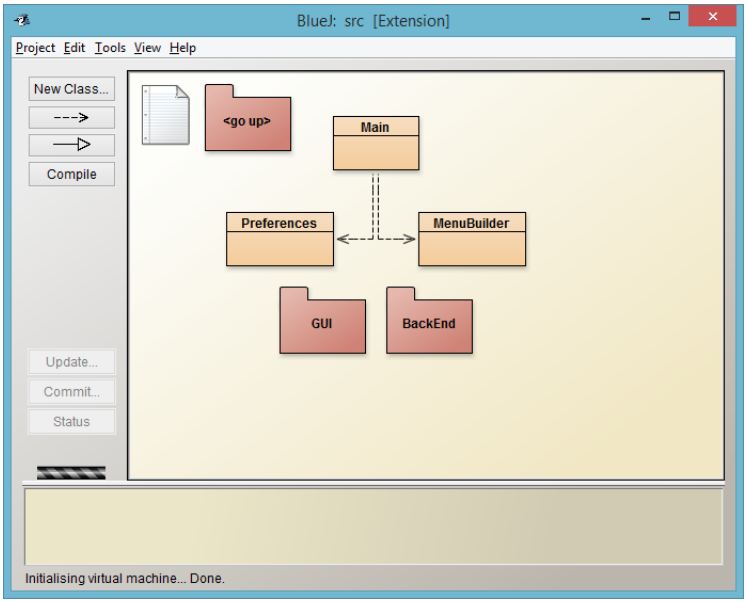
```
                        examples += "    " + examp.get(x) + "\n";
                    examplesLabel.setText(examples);
                }
            });

    }
}
```

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

Note: 1 artifacts were ommitted in this abridged version.

BlueJ

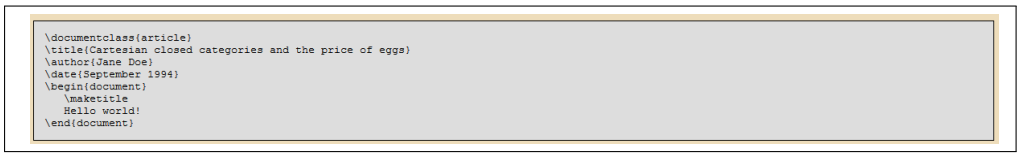
Description The following artifact is a screenshot of the BlueJ integrated development environment (IDE). BlueJ was our teams main development platform and was used early our development to build our extensions. BlueJ is also the platform we developed our extensions for.



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.