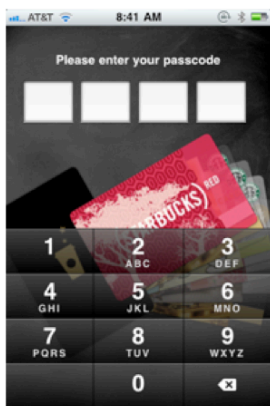
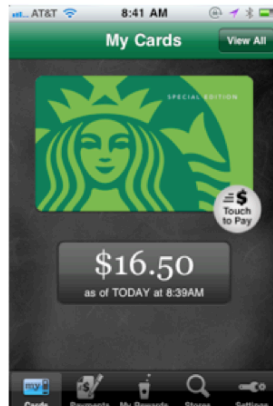


Individual Portion (100 Points)

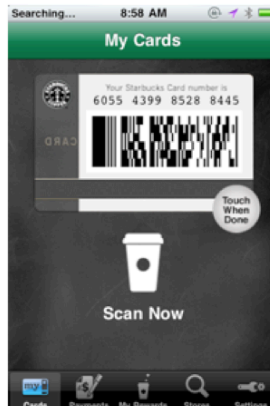
In this project, you will complete a partial implementation of the Patterns in a text based **Starbucks Mobile Apps Simulator**. This portion is a personal project and each team member will implement their own solution. Do not share or discuss your code and design (include your own unit tests) with other students. Doing so will be flagged by the grading system as plagiarism and result on a Zero score on the project. A personal "private" GitHub Repo will be assigned to you and the work in this Repo should be kept separate from your Team GitHub Repo.



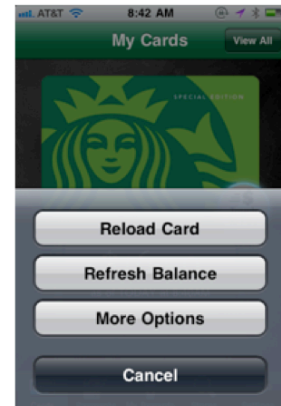
Pin Screen



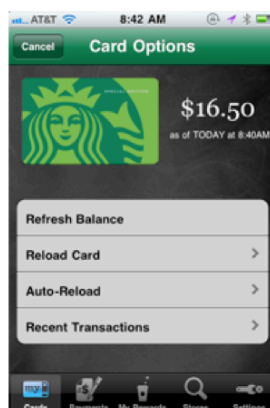
My Cards - Main



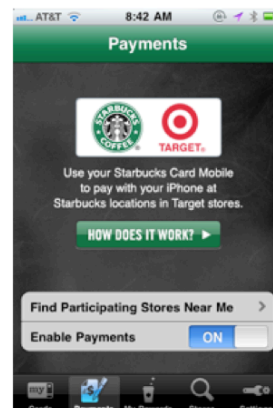
My Cards - Pay



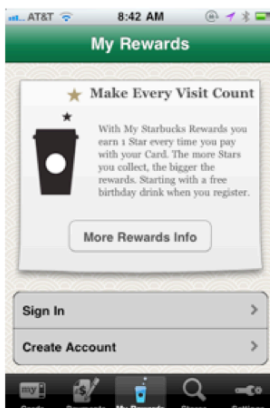
My Cards - Options



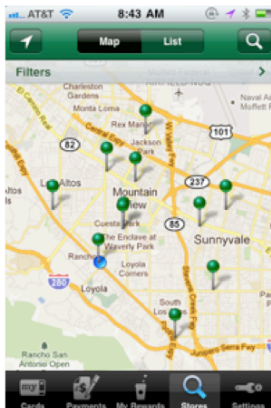
My Cards
More Options



Payment Setup




Rewards Setup



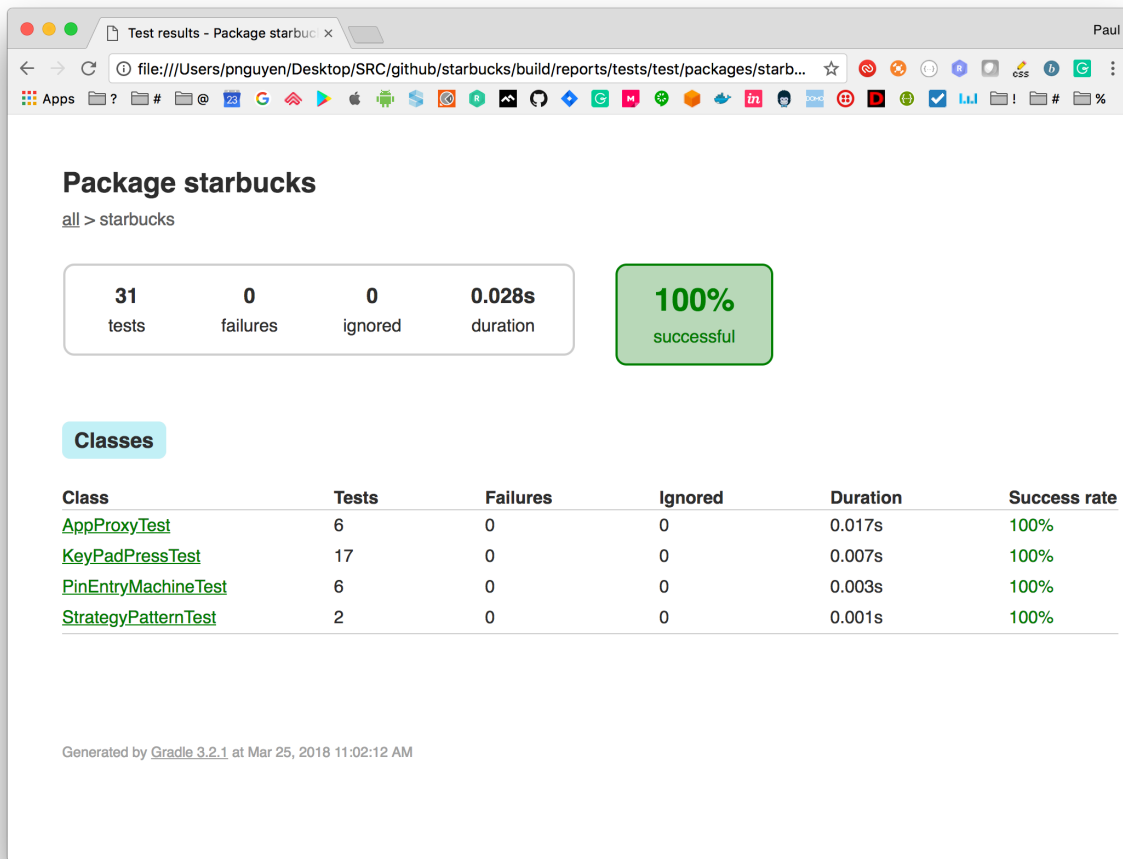
Find Starbucks

Requirements

- The initial code for the project should be in your assigned **Personal GitHub Repo** in the **starbucks** folder.
- *Please do not rename or re-organize this folder as doing so may impact **Grader** in processing your source code.*
- You will also need **Gradle**: <https://gradle.org/docs/#getting-started> ([Links to an external site.](#))[Links to an external site.](#)
- Review the set of **Requirements** and the starting set of Unit Tests in the Project
 - [Starbucks Mobile App - Project Requirements 2018.pdf](#) 
- Complete the implementation of this Project, such that your solution **passes all the tests** developed by TAs & Instructor.
- You will not be able to see the code for these tests; only the test results which will be posted in your **GitHub Repo**.
 - The **Grader will run tests and check-in the results to the directory starbucks/grader** in your *GitHub Repo*.

Test Report

- Using Gradle, you can run local unit tests and see test results in Gradle Build Artifacts. The test report will be available in the builds folder. For example (the initial state of the test report should look as follows for the provided starting test suite):



Unit Tests

- Sample initial Unit Tests are available here: <https://github.com/gopinathvinodhsjsu/starbucks/tree/master/src/test/java/starbucks> (Links to an external site.)Links to an external site.
- You may add your own tests into this folder in your Repo as you complete the implementation of the requirements
- The *Grader's test results (which is separate from your own unit tests)* will be added to your Repo in the "grader" folder.

UML Diagrams & Read Me Page

- **Maintain a UML Class Diagram and a Sequence Diagram along with Design notes on your approach on the Repo's README page in Markdown format.**
- **The Class Diagram context should focus only on the relevant classes for the "Add Card Screen".**
- **The Sequence Diagram should show the Object Collaborations for Adding a New Card.**
- **For UML diagrams, it is recommended that you switch to the Professional Version of Astah for this Project as it will allow you to reverse engineer the Java code to keep the diagrams in sync. A free Student License is available here: <http://astah.net/student-license-request> (Links to an external site.)Links to an external site..**

Grading

- **50 Points**
 - Will be based on the number of **Grader Tests your solution passes out of 100 minus 50**. The initial code provided should pass 50% of the test in the Grader test suite. Your goal is to increase the test rate but also to avoid failing past tests (i.e. regressions) with your changes.
- **50 Points**
 - Will be graded on **a Design and Code review of your work and your "coding habits" on the project. These will be based on:**
 - **Design** - Your **Design Notes and UML Diagrams** as provided in the README Markdown document in GitHub.
 - **Code Quality** - Your Code Quality, Choice of **Design Patterns, Approach to implementation** and "Code Smells" (or lack thereof - i.e. **How "Clean" is your code?**)
 - **Coding Habits** – This will be based on the ***Frequency and Quality of commits to the project Github***.
 - As such, it is expected that all contributions must be visible via Github. See the following guidelines for how GitHub counts contributions: <https://help.github.com/articles/why-are-my-contributions-not-showing-up-on-my-profile/> (Links to an external site.)Links to an external site.

Submission:

- **No Submission to Canvas is required**

- All work will be committed to GitHub
- **NOTE: At the end of class, your Personal project Repo will be kept private in the Course GitHub Account. Please do not post your solution on your own GitHub Repo visible to the Public. Doing so will violate the "Honor Code" as future students may plagiarize your solution.**

Team Portion (50 Points + 20 Points Extra Credit)

Your team will extend from the **Personal Project** and build a full end-to-end system for taking **Starbucks Orders**. The emphasis here is on team collaboration, so the points awarded will be based on individual contributions to the team and how the team performed overall. Extra credit will be given for additional challenges as spelled out below.

Suggested Project Components and Extra Credit (don't limit to these):

- **Components (one component per team member)**
 - **Domain Component** - Convert Personal Project solution into a Library and package as a JAR. This should be a "**business logic focused component**" such that **presentation and persistence should be plug-ins!**
 - *Note: do not manage the source code for this component in the Team's Repo*
 - **Add Cards API** - Implement a REST API on top of the Starbucks Library to enable Adding Cards
 - **Managed Order API** - Implement a REST API on top of the Starbucks Library to Manage Orders
 - **Payment API** - Implement a REST API on top of the Starbucks Library to enable Payments
 - **Other APIs** - Such as Authentication, User Profiles, Etc...
 - **Mobile App Simulator** - Implement an interactive UI using Processing or Greenfoot calling REST APIs

- **Extra (can be worked on by more than one team member):**
- **NOTE:** *Each extra credit option below is worth 10 points each. Select a max of two!*
 - Implement a "real" **iOS or Android Mobile App** calling the REST API
 - Deploy API to **AWS in an Auto Scaled EC2 Cluster** with Load Balancer
 - Deploy API to AWS as Docker Containers in **Amazon Containers**
 - Deploy API to AWS as Docker Containers in **Amazon EKS**
 - Implement **Web "Front-End" Deployed to Heroku** for Starbucks Payment Card management

Individual Requirements

- Select and **own one of the component** in the Team project.
- Keep a **Project Journal on GitHub** in markdown format to include:
 - **Weekly Scrum Report** (i.e. weekly version of daily scrum) which answers the tree daily stand-up questions:
 - What tasks did I work on / complete?
 - What am I planning to work on next?
 - What tasks are blocked waiting on another team member?
 - Select one of the **XP Core Values** and keep a journal of how the team kept these values throughout the project. Report this in your Project Journal with the weekly Scrum Report submissions:
 - Communication
 - Simplicity
 - Feedback
 - Courage
 - Respect
- **Maintain Weekly Scrum Task Board (in GitHub as a Project Board)**
 - Update the **Story** on your Task Board
 - Keep track of **remaining effort (hours)** and progress on a Team Task Board.
 - Use (and modify) the **Google Task Sheet** Template at:
 - Click on this [LINK \(Links to an external site.\)Links to an external site.](#). (Make adjustments to fit your team size)
 - Track your Team's **Burndown Chart** in this Sheet.

Team Requirements (This work should be done collaboratively -- i.e. Team ownership)

- Maintain the project in an assigned **Team GitHub Repo**.

- Maintain a **Scrum Task Board (using GitHub Project Board)**
- Create **UI Wireframes**
 - Create UI wireframes for each of the screen in your team's solution
 - (this can be done by hand or electronically with a tool like "Pencil")
- Create an Overall **Architecture Diagram (this does not have to be in UML)** showing:
 - Software Components and their Public Interfaces
 - The Dependencies between Components
 - The Relative Relationship of how these components are Deployed
 - Recommendation: Use UML Deployment/Component Diagram Notation.
 - <http://agilemodeling.com/artifacts/deploymentDiagram.htm> (Links to an external site.)Links to an external site.
 - <http://agilemodeling.com/artifacts/componentDiagram.htm> (Links to an external site.)Links to an external site.
- Maintain a README markdown file in the Team's GitHub Repo.
 - Include all **Diagrams, Design decisions** and the overall **Feature Set** of the project
- Perform **Project Demo**
 - Give a demo of your teams working prototype on "Demo Day"

Grading:

- Teams will be be graded with a **Team Score during Demo Day.**
 - **50 Points Max from Requirements**
 - **20 Points Max from Extra Credit**
- Individual deductions will be made to the Team Score based on contributions to be judged by:
 - **Completeness and Functioning Demo of your Component (as noted on Demo Day)**
 - **Frequency and Quality of commits** to the project Github.
 - As such, it is expected that all contributions must be visible via Github. See the following guidelines for how GitHub counts contributions: <https://help.github.com/articles/why-are-my-contributions-not-showing-up-on-my-profile/> (Links to an external site.)Links to an external site.

Submission (One per Team):

- **Your Team Name**
- **The names of each team member**

- A summary of areas of contributions (for each team member)
- Link to your team's GitHub Repo
- Link to your team's Project Board (on GitHub)
- Link to your team's Project Journal (on GitHub)
- Link to your team's Google Sprint Task Sheet

(*) At the end of the semester, the team's Repo will be released for Public access visible on this GitHub Account.

Example Format for Weekly Stand-up (i.e. Daily Scrum) and Final Burn-down Chart & Task Board:

Daily Scrum + Burndown Chart

Team Name, Sprint #1

Team Member Name

John Smith

What I did since the last daily scrum:

- Draw UML Class Diagram (done)
- Draw Sequence Diagram (not done, est. 2 more hours)

What I plan to do today:

- Draw Sequence Diagram
- Write Unit Tests

What blockers I have:

- I am waiting on the interface definition for my FooBar class. We need to define this ASAP.

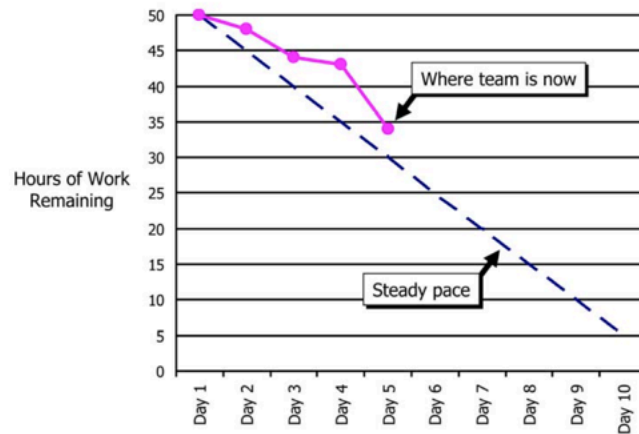


Figure 6. Burndown Chart

Backlog Item	Task	Task Owner	Initial Estimate	Hours of Work Remaining on Each Day of the Sprint									
				Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Enable all users to place book in shopping cart	Design business logic	Sanjay	4	4	3	3	1	0					
	Design user interface	Jing	2	2	1	1	1	1					
	Implement back-end code	Philip	6	6	2	5	2	0					
	Implement front-end code	Tracy	4	4	3	2	2	2					
	Complete unit testing	Sarah	4	4	3	3	3	3					
	Complete regression testing	Sarah	2	2	3	3	3	3					
	Write documentation	Sam	3	3	4	2	0	0					
Upgrade transaction processing module (must be able to support 500 transactions /sec)	Merge DCP code and complete layer-level tests	Jing	5	5	2	2	1	0					
	Complete machine order for pRank	Jing	4	4	2	0	0	0					
	Change DCP and reader to use pRank http API	Tracy	3	3	3	2	2	2					
Total				50	50	48	44	43	34				