Project Title
A Java API for unifying ad-hoc Wifi networking

Team Members
Klaus Cipi - kcipi2015@my.fit.edu
Peter Banis - pbanis2015@my.fit.edu
Michael Kolar - mkolar2015@my.fit.edu
Robert Olsen - olsenr2015@my.fit.edu

Faculty Sponsor
Dr. Marius Silaghi - msilaghi@fit.edu

Client
Dr. Marius Silaghi - Associate Professor at College of Engineering & Science, FIT

Meetings with Sponsor/Client
Friday, October 12
Monday, October 15
Tuesday, October 23

Progress of current Milestone (progress matrix)

Task	To Do (Completion %)	Peter	Klaus	Michael	Robert
Divorce the existing source code from DD (Direct Democracy) app	None (100% Complete)	25%	25%	25%	25%
Implement OS detection	None (100% Complete)	20%	50%	15%	15%
Create new Exceptions (Integrity check)	None (100% Complete)	40%	30%	15%	15%
Implement logging for the API	None (100% Complete)	50%	20%	15%	15%
Created configuration class for each OS	None (100% Complete)	25% (Linux)	25% (Mac)	25% (Windows)	25% (Windows)
Integration and Unit testing	None (100% Complete)	25%	25%	25%	25%

Discussion (at least a few sentences, ie a paragraph) of each accomplished task (and obstacles) for the current Milestone

- Divorce the existing source code from DD (Direct Democracy) app: We took the existing initialization scripts from the DD library and extended them. The existing scripts were designed for one application and were not modular (for example, you could not change the SSID of a network). These scripts were reworked or rewritten to be more useful to API consumers.
- Implement OS detection: We created a class which would detect what operating system the user was using based on the Java System.getProperty() method. This is useful because we later utilized this in our general superclass SystemConfiguration. A user only needs to create a SystemConfiguration object, and the API will automatically handle OS detection in order to find which subclass the user requires as their object instance.
- Create new Exceptions (Integrity check): The installation verifier will attempt to run the scripts suited to the operating system successfully. Along the way, there is always the possibility of the scripts not being found (IOException) or the validator simply not running to completion (InterruptedException). In addition to these pre-existing exceptions, we had to make a few of our own to cover more cases, namely UnknownOSException and ScriptFailureException. We must account for when the verifier cannot identify the operating system, and when the scripts cannot halt with the desired exit code.
- Implement logging for the API: When an exception occurs during the validation process, it is reported to the console and logged into the file “AdhocAPI.log”. Since it uses the packages in java.util.logging, the errors reported in the log file are familiar to the exceptions printed to stderr. The logger reports information such as the severity, thread, and location in code where the exception occured.
- Created configuration class for each OS: The user can create a SystemConfiguration object which allows them to create, disconnect, connect, and get various properties of a network by using its methods. The point of this object is that the user can create it and use it regardless of which operating system they’re using. At this point the supported operating systems are Windows, Mac OS, and Linux. Essentially, SystemConfiguration is an abstract super-class and all the methods are implemented in the subclasses SystemConfigurationX where X refers to the particular operating system. There is a method that allows the user to create a generic SystemConfiguration, but the method will automatically detect the user operating system and return a specialized instance of SystemConfiguration.
- Integration and Unit testing: Each individual method was tested. Afterwards entire classes were given a series of tests. For example, we tested each of our SystemConfiguration subclasses to make sure they worked. Finally, general tests which would use the general SystemConfiguration were created and employed.

Discussion (at least a few sentences, ie a paragraph) of contribution of each team member to the current Milestone

- Peter Banis: Peter was the main contributor to the Linux portion of the milestone. He looked into the functionality of the pre-existing scripts and researched modern approaches to refine the script set for Linux. He wrote at least half of the code for the logger across all operating systems, and helped write the new exceptions as well as the exception handling system across all operating systems. His testing was performed on Linux.
- Klaus Cipi: Klaus was the main contributor to the Mac portion of the milestone. Due to the incomplete and incompatible nature of what was already provided, Klaus had to start from scratch and write the Mac scripts in Swift. He helped Peter in making the Logger work and exceptions with Mac and Windows. His testing was performed on Mac.
- Michael Kolar: Michael worked together with Robert to complete the Windows part of this milestone. Michael implemented the methods in SystemConfigurationWindows.java, and he wrote the new windows scripts. He helped create the SystemConfigurationFactory.java class. Finally, he tested and troubleshooted all of these contributions.
- Robert Olsen: Robert helped to troubleshoot various smaller issues in the Windows scripts, and the logger, exception handling, and OS detection across all platforms. He helped in testing for the Windows platform using Windows 7. He wrote at least half the evaluation.

Plan for the next Milestone (task matrix)

Task	Peter	Michael	Klaus	Robert
Divorce API Networking code from Direct Democracy application	25%	25%	25%	25%
Expand and create networking functions	25%	25%	25%	25%
Acquire DirectP2P capable adapters and Android phones for future testing	25%	25%	25%	25%
Implement support for configuring devices in DirectP2P mode	25%	25%	25%	25%
Create more specific exceptions	25%	25%	25%	25%
Debug and correct connection problems between Windows 7/10 and Mac/Linux	25%	25%	25%	25%

Discussion (at least a few sentences, ie a paragraph) of each planned task for the next Milestone

- Divorce API Networking code from Direct Democracy application: When the code repository was given to us, it contained broken code that we couldn't use. Under Dr. Silaghi's recommendation we have been chiseling away by removing old files from the project and replacing them with our own work. So far we have removed the old scripts from the project and implemented our own. Now we need to finalize this process by scraping the old java files and creating our own implementations.
- Expand and create networking functions: We have gotten some of the basic functionality down where users can create, connect, and other obvious needs for networks. We will further our code base by also creating additional functions which are within the standard networking requirements and also offer additional utility to users.
- Acquire DirectP2P capable adapters and Android phones for future testing: We will access the requirements necessary for usable adapters. Furthermore, due to the tight financial constraints of a college-student lifestyle, we must make the most of our resources. First, we do background research into the market in order to get a baseline of price, quality, and discover noteable brands. Afterwards, we will be in a position of power while weighing our options. Each option will be considered by a price-benefit analysis with minor attention to aesthetics.
- Implement support for configuring devices in DirectP2P mode: Determine whether a given network interface supports the DirectP2P standard, and if it does allow that interface to be configured to use DirectP2P mode.
- Create more specific exceptions: Currently all exceptions are under the ScriptFailureExeption. We will add in other types(such as ScriptNotFoundException) to be more specific and helpful to developers using the API.
- Debug and correct connection problems between Windows 7/10 and Mac/Linux: Currently, Windows is capable of connecting to Windows and Linux networks. In terms of connecting to a Windows network, Windows and Mac are able to do so. The issue lies in how Windows doesn't allow its hosted networks to be ad-hoc. Instead it causes the computer to be used as a pseudo-router which makes it an infrastructure network. This is why is doesn't register as ad-hoc by Linux and thus Linux can't connect to the Windows hosted network. Windows 10 has removed all of its ad-hoc capabilities.

Sponsor feedback on each task for the current Milestone

- Divorce the existing source code from DD (Direct Democracy) library:
- Implement OS detection:
- Create new Exceptions (Integrity check):
- Implement logging for the API:
- Created configuration class for each OS:
- Integration and Unit testing:

Sponsor Signature: ________________________________ Date: ________

Sponsor Evaluation

- Sponsor: detach and return this page to Dr. Chan (HC 322)
- Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Peter Banis	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Klaus Cipi	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Michael Kolar	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Robert Olsen	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

- Sponsor Signature: ________________________________ Date: ________