Syllabus

Course Overview

Software construction is the practice of organizing the design and development of software, the core of information technology that is fundamental to our personal and professional lives. This course is focused on the processes and tools used to develop software applications. These processes and tools can include:

- Programming Languages.
- Software Construction Techniques.
- Software Development Tools.
- Testing.
- Maintenance.

Emphasizing modern software engineering approaches such as Agile development, the learner will create and test software while being introduced to software construction as a process. The core concepts learned in this class are relevant across all modern programming languages and software development projects.

In this course you will construct a software application to operate a Smart Thermostat using Java.

Course Competencies

(Read Only)

To successfully complete this course, you will be expected to:

1. Apply software construction practices.
2. Conduct life cycle software construction tasks.
3. Evaluate software construction technologies and methods.
4. Apply software construction tools.
5. Justify coding and design decisions.

Course Prerequisites

IT4772

The following required readings are provided in the Capella University Library or linked directly in this course. To find specific readings by journal or book title, use Journal and Book Locator. Refer to the Journal and Book Locator library guide to learn how to use this tool.

- Stephens, R. (2015). *Beginning software engineering.* Hoboken, NJ: Wiley.

External Resource

Please note that URLs change frequently. While the URLs were current when this course was designed, some may no longer be valid. If you cannot access a specific link, contact your instructor for an alternative URL. Permissions for the following links have been either granted or deemed appropriate for educational use at the time of course publication.

- Agile Modeling. (n.d.). Agile requirements change management. Retrieved from http://agilemodeling.com/essays/changeManagement.htm
- Agile Modeling. (n.d.). User Interface (UI) prototypes: An Agile introduction. Retrieved from http://agilemodeling.com/artifacts/uiPrototype.htm
- Agile Modeling. (n.d.). User stories: An Agile introduction. Retrieved from http://www.agilemodeling.com/artifacts/userStory.htm
- Agile Trail. (2015). Sprint review, a feedback gathering event: 17 questions and 8 techniques. Retrieved from http://agiletrail.com/2015/09/19/sprint-review-a-feedback-gathering-event-17-questions-and-8-techniques/
- Ambysoft. (n.d.). Agile testing and quality strategies: Discipline over rhetoric. Retrieved from http://www.ambysoft.com/essays/agileTesting.html#AgileSoftwareDevelopment
- App Partner. (2017). 10 reasons why Agile is best for app development & how we use it. Retrieved from https://medium.com/app-partners-how-to-start-your-app-startup/10-reasons-why-agile-is-best-for-app-development-how-we-use-it-36c53bd0cf9c
- Atlassian. (n.d.). Learn Git with Bitbucket Cloud. Retrieved from https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud
- Atlassian. (n.d.). Setting up a repository. Retrieved from https://www.atlassian.com/git/tutorials/setting-up-a-repository
- Axosoft. (2017). How to use fast feedback loops. Retrieved from https://blog.axosoft.com/feedback-loops-agile-development/
- Baeldung. (2017). JAX-RS client with Jersey. Retrieved from http://www.baeldung.com/jersey-jax-rs-client
- Bitbucket. (2017). Bitbucket. Retrieved from https://bitbucket.org/
- Chron. (n.d.). How do I write a standard operations procedures manual? Retrieved from http://smallbusiness.chron.com/write-standard-operations-procedures-manual-2596.html
- Continuous Dev. (2015). Checkstyle vs PMD vs Findbugs. Retrieved from http://continuousdev.com/2015/08/checkstyle-vs-pmd-vs-findbugs/
- Designmodo. (2016). Wireframing, prototyping, mockuping - What's the difference? Retrieved from https://designmodo.com/wireframing-prototyping-mockuping/
- Dice. (2015). Choosing an IDE that’s right for you. Retrieved from https://insights.dice.com/2015/05/19/choosing-an-ide-right-for-you/
- DZone. (2017). An introduction to software architecture: What you should know. Retrieved from https://dzone.com/articles/an-introduction-to-software-architecture-what-you
- Envato Tuts+. (2017). Ensure high-quality Android code with static analysis tools. Retrieved from https://code.tutsplus.com/tutorials/ensure-high-quality-android-code-with-static-analysis-tools--cms-28787
- Guru99. (n.d.). Test-Driven Development (TDD): Learn with example. Retrieved from https://www.guru99.com/test-driven-development.html
- IBM developerWorks. (2011). 11 proven practices for more effective, efficient peer code review. Retrieved from https://www.ibm.com/developerworks/rational/library/11-proven-practices-for-peer-review
- Java World. (2016). Choosing your Java IDE. Retrieved from https://www.javaworld.com/article/3114167/development-tools/choosing-your-java-ide.html
- Jersey. (n.d.). Chapter 5. Client API. Retrieved from https://jersey.github.io/documentation/latest/client.html
- Joel on Software. (2001). User interface design for programmers. Retrieved from https://www.joelonsoftware.com/2001/10/24/user-interface-design-for-programmers/
- martinfowler.com. (n.d.). Agile software development. Retrieved from https://martinfowler.com/agile.html
- Oracle. (2013). Using Scene Builder with NetBeans IDE. Retrieved from http://docs.oracle.com/javafx/scenebuilder/1/use_java_ides/sb-with-nb.htm#CIHDHEFE
- Oracle. (n.d.). Class Properties. Retrieved from https://docs.oracle.com/javase/8/docs/api/java/util/Properties.html
- Oracle. (n.d.). File I/O. Retrieved from https://docs.oracle.com/javase/tutorial/essential/io/fileio.html
- Oracle. (n.d.). JavaFX: Working with JavaFX UI components. Retrieved from https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/index.html
- Oracle. (n.d.). JDK 8u151 with NetBeans 8.2. Retrieved from http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html
- Oracle. (n.d.). Lesson: Exceptions. Retrieved from https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html
- Oracle. (n.d.). Properties. Retrieved from https://docs.oracle.com/javase/tutorial/essential/environment/properties.html

- Prosci. (n.d.). Adapting and adjusting change management in an Agile project. Retrieved from http://blog.prosci.com/adapting-and-adjusting-change-management-in-agile
- RBCS. (n.d.). Segue [PDF]. Retrieved from http://rbcs-us.com/documents/Segue.pdf
- RBCS. (n.d.). Why most unit testing is waste [PDF]. Retrieved from http://rbcs-us.com/documents/Why-Most-Unit-Testing-is-Waste.pdf
- SubMain. (2016). Why automate code reviews? Retrieved from https://blog.submain.com/why-automate-code-reviews/
- The Holy Java. (2015). Challenging myself with Coplien's Why Most Unit Testing is Waste. Retrieved from https://theholyjava.wordpress.com/2015/01/26/
- ThoughtWorks. (2013). Providing 'just enough design' can make Agile software delivery more successful. Retrieved from https://www.thoughtworks.com/insights/blog/providing-just-enough-design-can-make-agile-software-delivery-more-successful
- ThoughtWorks. (2014). Using TDD to influence design. Retrieved from https://www.thoughtworks.com/insights/blog/using-tdd-influence-design
- Tutorialspoint. (n.d.). Software engineering overview. Retrieved from https://www.tutorialspoint.com/software_engineering/software_engineering_overview.htm
- UX Booth. (2015). User stories: A foundation for UI design. Retrieved from http://www.uxbooth.com/articles/user-stories-a-foundation-for-ui-design/
- UX Mastery. (2016). Wireframing for beginners. Retrieved from https://uxmastery.com/wireframing-for-beginners/
- UX Planet. (2016). Golden rules of user interface design. Retrieved from https://uxplanet.org/golden-rules-of-user-interface-design-19282aeb06b
- Wikibooks. (2011). Introduction to Software Engineering - Tools/IDE. Retrieved from https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Tools/IDE
- Wikibooks. (2012). Introduction to Software Engineering/Tools/Source Control. Retrieved from https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Tools/Source_Control
- Wikibooks. (2017). Introduction to Software Engineering/Process/Agile Model. Retrieved from https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/Agile_Mode
- WikiHow. (n.d.). How to create a user manual. Retrieved from https://www.wikihow.com/Create-a-User-Manual

Suggested

The following materials are recommended to provide you with a better understanding of the topics in this course. These materials are not required to complete the course, but they are aligned to course activities and assessments and are highly recommended for your use.

Optional

The following optional materials are offered to provide you with a better understanding of the topics in this course. These materials are not required to complete the course.

External Resource

Please note that URLs change frequently. While the URLs were current when this course was designed, some may no longer be valid. If you cannot access a specific link, contact your instructor for an alternative URL. Permissions for the following links have been either granted or deemed appropriate for educational use at the time of course publication.

- Eclipse. (n.d.). Eclipse IDE for Java developers. Retrieved from https://eclipse.org/downloads/packages/eclipse-ide-java-developers/oxygenr
- Git. (n.d.). Downloads. Retrieved from https://git-scm.com/downloads
- Google. (n.d.). Google Java style guide. Retrieved from https://google.github.io/styleguide/javaguide.html
- JetBrains. (n.d.). IntelliJ IDEA. Retrieved from https://www.jetbrains.com/idea/
- NetBeans. (n.d.). Writing JUnit tests in NetBeans IDE. Retrieved from https://netbeans.org/kb/docs/java/junit-intro.html
- Software Engineering Institute. (2015). Java coding guidelines. Retrieved from https://wiki.sei.cmu.edu/confluence/display/java/Java+Coding+Guidelines

Projects

Project >> Smart Thermostat

Project Overview

The course project requires you to construct software for a Smart Thermostat through a series of assignments. You will take an Agile approach to developing software defined in a prioritized Software Requirements Specification (SRS) document (and user stories) during a series of four Sprints. You are introduced to the project through notes taken from the project kick-off meeting.

Your role is that of a software developer who is using NetBeans for software development. During the course of your assignments you:

- Perform an IDE platform evaluation.
- Read and display temperature settings.
- Store and retrieve programmed settings.
- Develop the Operate Screen GUI.
- Develop the View Programs screen GUI.
- Conduct unit testing.
- Examine the merits of code reviews.
- Adjust to a changed user story.
- Create user operating instructions.

Unit 1 >> Software Construction Fundamentals

Introduction

The Software Engineering Body of Knowledge (SWEBOK Guide) identifies four major concepts in software construction fundamentals:

Minimizing Complexity

Studies have shown that people are limited in our ability to hold complex or meaningful information in our working memories, something along the lines of 7 plus or minus 2 (Miller, 1956). Therefore, any effort to minimize complexity during development is a strong driver in software construction. However, as additional functionality is added to the software development project, complexity increases. How do we minimize complexity but still add functionality as the project progresses in its development? There are a number of approaches, some structural such as coding style standards, others construction-oriented quality techniques using tools like FindBugs, PMD, and Lint to identify common bugs as the developer writes code. In this way, the developer does not have to think about basic code styling and avoiding basic coding errors. This frees the developer to focus more on the design of the code than the actual effort of coding.

Anticipating Change

Traditional software development methodologies such as waterfall or SDLC attempt to anticipate the inevitable change that comes with software development through such practices such as planning and coding for flexibility. However, quite often, change comes from a direction that is not anticipated. Therefore, developers have embraced new development methodologies which focus not on anticipating change but accommodating change through a series of standards and practices that accommodate that change. These standards, practices and techniques such as designing for change and not coding for future requirements that may not occur can be used to make software changes easier with less impact on other parts of the software system.

Constructing for Verification

As part of the daily coding tasks conducted by the developer, unit testing is performed. This test code is low-level, focusing on a particular piece of code. These unit tests are usually written using some unit testing framework and are designed to be fast in terms of execution as they are run very frequently. As the number of unit tests grow, developers will begin to organize the code and the tests into suites in order to both segregate functionality and the tests associated with that code suite and to organize the code to support automated testing.

Standards in Construction

While programming sometimes is considered an artistic task, developers must adhere to a number of standards. These standards are normally created and maintained by national and international governing bodies, or in rare cases, created by the firm. Standards exist for everything from the programming languages used by the developer (Java and C#), to the operating system and platform interface standards (J2EE or .NET), to the tooling standards (Docker or Unified Modeling Language). In the case of internal standards within a given firm, standards may exist that support the other three concepts such as minimizing complexity, anticipating change and constructing for verification. Normally, these standards are decided upon before construction and many are pre-determined based on such inputs as programming language or operating system. Based on the type of project or needs of the project, additional standards may be selected.

Reference

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*, 81–97.

Learning Activities

u01s1 - Studies

Readings

Read the following in *Beginning Software Engineering*:

- Chapter 1, "Software Engineering From 20,000 Feet," pages 3–14.

Use the Internet to complete the following:

- Dzone. (2017). An introduction to software architecture: What you should know. Retrieved from https://dzone.com/articles/an-introduction-to-software-architecture-what-you
    - Discusses the role of software architecture and software architect in software construction.
- Tutorialspoint. (n.d.). Software engineering overview. Retrieved from https://www.tutorialspoint.com/software_engineering/software_engineering_overview.htm
    - A basic introduction to software engineering.

Optional Readings

The following resources can be used as references to aid your understanding of Java coding and style best practices:

- Google. (n.d.). Google Java style guide. Retrieved from https://google.github.io/styleguide/javaguide.html
- Software Engineering Institute. (2015). Java coding guidelines. Retrieved from https://wiki.sei.cmu.edu/confluence/display/java/Java+Coding+Guidelines

u01s2 - Course Software

Software for This Course

Java NetBeans Cobundle

Download the JDK u8151 with NetBeans 8.2 file appropriate for your OS. Make sure to accept the license agreement first.

Other Software

IntelliJ IDEA & Eclipse

Review the following platforms in preparation for this unit's assignment:

- IntelliJ IDEA
- Eclipse IDE for Java Developers.

Source Code Control

This course requires Git client software as the source code control toolset. This supports a local repository. It is available 100% free of charge directly from the Git open source community. The Git client software consists of a small toolset available for Mac OS X, Windows and Linux. Download the software at: https://git-scm.com/downloads.

Note: In addition to the Git software, there are GUI clients for Git such as SourceTree (Max OS X and Windows) and TortoiseGit (Windows only). You can download them if you would feel more comfortable with a GUI tool rather than the command-line but using these GUI tools will be unsupported by the Capella Technical Support staff.

Remote Repositories

As a distributed source code control system, Git has both a local source code repository (located on the developer's local hard disk) and a remote repository that changes in the local repository are synchronized to. A number of vendors support Git remote repositories by offering a cloud-based hosting service for those remote repositories. Two of the most notable hosting services are GitHub and Bitbucket.

Note: For this class, you are expected to maintain your assignment code base in Git and sync their local repository to Bitbucket through 'pushing' their local repository changes to the remote repository.

u01s2 - Learning Components

- Review Eclipse, IDEA, and NetBeans platforms.

u01d1 - Software Engineering as a Discipline

If you look at the major themes of the SWEBOK Software Construction Fundamentals, none of them discuss programming languages. The idea of software construction is to build a software application with a certain level of quality while meeting the requirements of the stakeholder within the time frame and budgetary constraints.

Discuss how using the tenets of the SWEBOK Software Construction Fundamentals and a software development methodology enforces the practices of software engineering as a structured discipline and practice.

Your post should be approximately 250 words. Support and cite your discussion with research from online materials, books, and other resources.

Response Guidelines

Respond substantially to the post of at least one of your peers. Your response should be approximately 200 words or more. Some approaches include providing an alternative viewpoint, substantiating your peer's claims with referenced material, expanding on the noted topic by citing experience, or conducting a review of the literature to further the exploration of the subject. Where possible, you should elaborate on the topic based on your professional experience and try to tie the topic to real-life expertise.

Course Resources
Undergraduate Discussion Participation Scoring Guide

u01d1 - Learning Components

- Study fundamentals of software construction.

Unit 2 >> Software Development Tools

Introduction

One of the first decisions you will make as an independent developer on a small project or as a lead technical developer responsible for a team of programmers is which software to employ. It is tempting to choose an industry leader, going with the "best of breed" that is determined by some analytic group such as Gartner who track such things, but you and your team may end up paying for that decision when the product doesn't suit your or your team's style or approach to development. Rather than taking the path of least resistance and adopting the market leader by default, a more valuable exercise that will pay off in the long run is to work through the factors and features that should influence your software decision, evaluate the vendor's software offerings against those factors, and then determine which one best meets your needs.

Determining which software to use can be a complicated process. There are usually quite a number of choices and the vendor marketing material and sales representatives don't always make it known to you what your options are. In addition, software can be an expensive tool that is essential to your work. It can be expensive not only in money but in productive time needed to understand how the software will work in your environment. You need the right software for the task you and your team are undertaking, and you cannot afford to choose a tool that is not going to aid you in that effort.

The first step in that determination is a Software Product Evaluation. This effort goes through a number of steps in the evaluation and selection process. Some of these steps include determining business and technical needs, qualifying potential solutions, qualifying vendors who offer those potential solutions, conduct a software assessment against the business and technical factors, and aggregate the results into a recommendation.

Learning Activities

u02s1 - Studies

Readings

Read the following in *Beginning Software Engineering*:

- Chapter 2, "Before the Beginning," pages 16–25.

Use the Internet to complete the following:

- Java World. (2017). Choosing your Java IDE. Retrieved from https://www.javaworld.com/article/3114167/development-tools/choosing-your-java-ide.html
    - This article compares three leading Java integrated development environments (IDE) in terms of features and functionality.
- Dice. (2015). Choosing an IDE that's right for you. Retrieved from https://insights.dice.com/2015/05/19/choosing-an-ide-right-for-you/
    - The article covers IDE selection criteria.
- Wikibooks. (2011). Introduction to Software Engineering - Tools/IDE. Retrieved from https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Tools/IDE
    - This chapter discusses common integrated development environment capabilities found in most leading IDEs.
- Wikibooks. (2012). Introduction to Software Engineering - Tools/Source Control. Retrieved from https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Tools/Source_Control
    - This chapter discusses source code control, its role in software development and common functions.
- Atlassian. (2017). Learn Git with Bitbucket Cloud. Retrieved from https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud
- Atlassian. (2017). Setting up a repository. Retrieved from https://www.atlassian.com/git/tutorials/setting-up-a-repository

Capella Documents

Read the following documents related to your course project.

- SRS.
- Kick-off Meeting Notes.
- u03a1 User Story.
- u04a1 User Story.
- u05a1 User Story.
- u06a1 User Story.

u02d1 - Software Selection Templates and Methodologies

Share a software criteria selection template or methodology that you found on the web.

Discuss features that you believe are relevant to this project. What did you find useful? How might you use it in the course project?

Your post should be approximately 250 words. Support and cite your discussion with research from online materials, books, and other resources.

Response Guidelines

Respond substantially to the post of at least one of your peers. Your response should be approximately 200 words or more. Some approaches include providing an alternative viewpoint, substantiating your peer's claims with referenced material, expanding on the noted topic by citing experience, or conducting a review of the literature to further the exploration of the subject. Where possible, you should elaborate on the topic based on your professional experience and try to tie the topic to real-life expertise.

Course Resources

Undergraduate Discussion Participation Scoring Guide

u02d1 - Learning Components

- Review examples of a platform selection methodologies.

- Understand business requirements or restrictions for a development platform.
- Understand Platform requirements for a project.

u02a1 - Project Platform Evaluation

Overview

Most software development groups have a standard methodology and tools for a project that they customarily employ as a part of their business. Occasionally companies may reevaluate their tools or platforms for various reasons. Reasons companies might consider are the available templates, corporate standards, proprietary vs. open source, plug-ins, integration with tools and source controls, and so on. What will your criteria be?

In this assignment you evaluate Oracle NetBeans, Eclipse, and IntelliJ IDEA to see which might be a better fit for your project. You will also consider its implications in the scope of a larger smart home development project.

Preparation

- Read the SRS, Kick-off meeting notes, and each of the four user stories found in the Resources.
- Have the Java NetBeans Cobundle (downloaded in u01s2) available. This may help with your software evaluation effort.

Directions

Define 8–10 platform selection criteria for the project. Then, compare the Oracle NetBeans, Eclipse, and IntelliJ IDEA platforms and recommend which is the most appropriate for this project.

Make sure to do the following:

- Compare development platform features.
- Define appropriate platform selection criteria that effectively address project requirements.
- Recommend a development platform using your chosen criteria.

Additional Requirements

- Font: Times New Roman 12 point.
- Length: 3–4 double-spaced pages not including cover page and current APA compliant reference page.

Course Resources
u03a1 User Story
u04a1 User Story
u05a1 User Story
u06a1 User Story
Software Requirements Specification (SRS)
Kick-off Meeting Notes

Unit 3 >> Agile and User Stories

Introduction

The importance software requirements is self-evident, but experience has shown that defining a full set of software requirements early in a project often proves counterproductive. This is because it is not possible to define all the detailed requirements at the outset of a software development project of any length.

One important thing to keep in mind is that the business environment the software will support undergoes changes as the software development timeline progresses. New requirements and opportunities present themselves as the development team attempt to address the requirements that were documented. In addition, as the project progresses both the business stakeholders responsible for defining the requirements and the development team understand more about the business needs and how the software can best support the business.

Too many times, defining detailed software requirements too early in the development process (usually in Requirements Analysis) will mean that those requirements specifications will need to be altered in mid-construction resulting in a certain portion of the work being discarded or delivering software that meets the originally-specified requirements but subsequently failing to satisfy the business needs adequately.

Acknowledging this fact, the Agile approach to software development proposes a better way of working. The software development team would still capture software requirements, but at a higher level earlier in the project. Further detail is elicited as the project progresses, these requirements are defined as a User Story. A User Story is a requirement expressed from the perspective of an end-user goal with little detail as not to constrain the solution. It is a well-understood and popular approach for expressing software requirements within the Agile methodology. The strengths of this approach include:

- The User Story focuses on the viewpoint of a user who will be impacted by the solution.
- The User Story defines the requirement in language that a user would readily understand.
- The User Story brings clarity to the reasons that the requirement exists.
- The User Story helps define high level requirements without low level detail.

Learning Activities

u03s1 - Studies

Readings

Use *Beginning Software Engineering* to complete the following:

- In Chapter 4, "Requirement Gathering," read "User Stories," page 77.
- In Chapter 14, "RAD," read "Agile," pages 307–316.

Use the Internet to complete the following:

- Wikibooks. (2017). Introduction to Software Engineering/Process/Agile Model. Retrieved from https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/Agile_Model
    - An introduction to the Agile software development methodology.
- Agile Modeling. (n.d.). User stories: An Agile introduction. Retrieved from http://www.agilemodeling.com/artifacts/userStory.htm
    - An introduction to user stories, Agile's requirements definition approach.
- Martinfowler.com. (n.d.). Agile software development. Retrieved from https://martinfowler.com/agile.html
    - Teachings on using Agile in software development.
- Baeldung. (2017). JAX-RS client with Jersey. Retrieved from http://www.baeldung.com/jersey-jax-rs-client
    - Using Jersey and Java J2EE JAX-RS to build a Java client driver class that can call a web service.
- Jersey. (n.d.). Chapter 5. Client API. Retrieved from https://jersey.github.io/documentation/latest/client.html
    - A more in-depth look at Jersey (an implementation of Java JAX-RS) and the Client API. Start with 5.3, Overview of the the Client API.

u03d1 - Requirements, User Stories & the Design Process

Describe the basic elements of the software development life cycle. Why is an understanding of the life cycle relevant for your understanding of software construction? Supply evidence to support your assertions.

Your post should be approximately 250 words. Support and cite your discussion with research from online materials, books, and other resources.

Response Guidelines

Respond substantially to the post of at least one of your peers. Your response should be approximately 200 words or more. Some approaches include providing an alternative viewpoint, substantiating your peer's claims with referenced material, expanding on the noted topic by citing experience, or conducting a review of the literature to further the exploration of the subject. Where possible, you should elaborate on the topic based on your professional experience and try to tie the topic to real-life expertise.

Course Resources
Undergraduate Discussion Participation Scoring Guide

u03a1 - Read and Display Temperature Settings (Sprint 1)

Overview

A typical program rarely exists by itself and usually relies on network communications. Data retrieval requests are common in modern software applications. The use of a web service is a very common mechanism for facilitating this type of communication.

NetBeans is a commonly used IDE to develop Java applications. You will use it during the remainder of your assignments to develop your Smart Thermostat application called ProgrammableThermostat.

Scenario

You are assuming the role of a software developer and are beginning work on the first of four project Sprints. The first step in developing the application is defining some of the classes upon which your application will depend.

For this assignment, you begin building the ProgrammableThermostat application by creating Java classes that query a Web Service and display temperature data.

Preparation

- Make sure that you have downloaded and Installed NetBeans per the instructions in u01s2.
- Create a project called "ProgrammableThermostat" in NetBeans.
- Review the temperature.json file found in the Resources.
- Read the u03a1 User Story found in the Resources.
- Your code will need to read temperature data stored in JSON format using a web service to access the JSON information stored on a Capella web server. You will create a web query to retrieve the data. The result of the query will be returned as JSON data. To view the JSON data directly you can paste the URL (found in the Resources) that tells where the data is available into a browser.
    - **Note:** browsers usually do not display JSON information well as they do not understand how to format the data. You may want to investigate using a browser add-on (depending upon your browser) that understands and formats JSON. JSONView is a possibility – it works for several browsers. (Paste "JSONView" into your favorite search engine for details.)
- **Important Note:** For this class, you are expected to maintain your assignment code base in Git and sync their local repository to Bitbucket through 'pushing' their local repository changes to the remote repository. Make sure you have created a Bitbucket account. A link to it can

be found in the Resources.

Directions

Complete both parts of this assignment.

Part 1 – Create Classes and Build the Display Driver

Open your ProgrammableThermostat project in NetBeans, then write Java classes that employ standard Java coding conventions for building the display driver specified in the u03a1 User Story:

Do the following:

- Develop Java networking classes/methods that read the temperature data from the web service.
- Develop Java classes/methods that convert the raw temperature data into Java objects.
- Build a driver to display the temperature data on the Java Console. Note the following:
    - You do not need to create a GUI; your data should be displayed in the NetBeans Java Console.
    - Take a screen shot of the completed Java Console to submit with your assignment.

Part 2 - Justify Design and Coding Decisions

Write at least three paragraphs that document and describe another significant coding approach that you could have used to accomplish the same results and functionality for retrieving and displaying the data in Part 1. Justify why you chose your approach over the alternative you described. Do this in a separate word document to submit within your ZIP file.

Submission Requirements

You must upload your ProgrammableThermostat files to Bitbucket, then submit a single ZIP file in the courseroom named u03a1.zip. It should contain:

- The NetBeans ProgrammableThermostat project directory.
- The screen shot of the runtime output when your program runs.
- The Word document containing the content from Part 2 in the Directions.

Course Resources
temperature.json
Bitbucket
u03a1 User Story

Unit 4 >> Software Construction Using Agile

Introduction

Looking at software development from the perspective of the developer, development has evolved into performing a series of short, repeated development cycles based on a set cycle, typically two weeks. Using this iterative software development cycle to build a software application incrementally is one of the key principles of Agile software development. This principle of using short, rapid, iterative steps is crucial to reviewing and validating user requirements. The term "iterative" refers to a pattern of tasks that a software developer follows in repeated cycles (one cycle per user story) for a fixed duration, normally a development sprint. Through this process, the software development team becomes "agile" whereby changes in requirements drive future iterations based on stakeholder feedback.

In these two-week sprints, the software developer performs a number of tasks for each user story they are building. These tasks include:

- **Design:** incremental design (Martin Fowler calls this "evolutionary" design) using software patterns and third-party frameworks to drive design of the software created for the user story.
- **Software development:** actual software programming of the functionality documented in the user story.
- **Unit testing:** short cycles of development require good unit tests in order to ensure that new code does not break previous work. In addition, these unit tests are a form of system documentation as it demonstrates how the application should function.
- **Integration:** checking the software code into the source code repository. Once checked it, the software developer executes integration tests to ensure that their code submission did not cause issues with the software application as a whole and that the software applications continues to function correctly.

All modern programming languages provide various constructs both to read from and write to files, usually located on persistent data storage, either a server hard drive or some type of network attached storage. In the case of Java, there are standard ways of reading from and writing to files. Many references still exist that point at the original java.io package, however, modern Java applications use the more recent java.nio.file API. This is the API you should use in your application.

Learning Activities

u04s1 - Studies

Readings

Read the following in *Beginning Software Engineering*:

- Chapter 7, "Development," pages 155–169.

Use the Internet to complete the following:

- App Partner. (2017). 10 reasons why Agile is best for app development & how we use it. Retrieved from https://medium.com/app-partners-how-to-start-your-app-startup/10-reasons-why-agile-is-best-for-app-development-how-we-use-it-36c53bd0cf9c
  - Discusses why Agile is the preferred approach to software development.
- Ambysoft. (n.d.). Agile testing and quality strategies: Discipline over rhetoric. Retrieved from http://www.ambysoft.com/essays/agileTesting.html#AgileSoftwareDevelopment
  - An in-depth look at Agile software development and testing.
- Oracle. (n.d.). File I/O. Retrieved from https://docs.oracle.com/javase/tutorial/essential/io/fileio.html
  - An overview on Java file management and reading/writing files to local storage.
- Oracle. (n.d.). Lesson: Exceptions. Retrieved from https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html
  - Overview of Java Exception Handling.
- Oracle. (n.d.). Properties. Retrieved from https://docs.oracle.com/javase/tutorial/essential/environment/properties.html
  - An overview on the Java Properties class, a key/value storage mechanism for configuration parameters.
- Oracle. (n.d.). Class Properties. Retrieved from https://docs.oracle.com/javase/8/docs/api/java/util/Properties.html
  - The Java Properties class JavaDoc documentation.

u04d1 - Persistent Data

There are a number of approaches to making your applications' information persistent. This includes saving program information to a database table or to local storage either through serialization or through more basic text files (such as the Java Properties class).

Discuss the advantages and disadvantages to each approach, the approach you would recommend, and why you would recommend that approach.

Your post should be approximately 250 words. Support and cite your discussion with research from online materials, books, and other resources.

Response Guidelines

Respond substantially to the post of at least one of your peers. Your response should be approximately 200 words or more. Some approaches include providing an alternative viewpoint, substantiating your peer's claims with referenced material, expanding on the noted topic by citing experience, or conducting a review of the literature to further the exploration of the subject. Where possible, you should elaborate on the topic based on your professional experience and try to tie the topic to real-life expertise.

Course Resources

Undergraduate Discussion Participation Scoring Guide

u04d1 - Learning Components

- Perform Java I/O using the NIO API.
- Understand principles of error handling.

u04a1 - Store and Retrieve Programmed Temperature Settings (Sprint 2)

Overview

As the Homeowner, you want the ability to program your thermostat to engage the furnace/air conditioner as necessary to accommodate the indoor temperature setting. Once the Homeowner has programmed the settings for a given day/hour, the inputs need to be stored so that ProgrammableThermostat can reference them.

In this assignment (Sprint 2), you will create classes that read and write to files that contain the programmed temperature settings for any given day/hour.

Preparation

- Read the u04a1 User Story found in the Resources.

Directions

Part 1 - Add Java Classes

Add Java classes to your ProgrammableThermostat project that read/write programmable temperature settings and times to disk. Include appropriate file I/O error handling in your code that addresses issues such as "file not found" and other typical errors.

Do the following:

- Create a Java class with methods that read and write user programmed settings to and from a file.
- Write appropriate file I/O error handling to identify code errors.

Take screen captures of your program console that shows that your temperature settings have been created and saved. You will submit them with your assignment.

Part 2 - Justify Design and Coding Decisions

Write at least three paragraphs that justifying your input/output error handling methodology. Do this in a separate word document and submit it within your ZIP file.

Submission Requirements

You must upload your ProgrammableThermostat files to Bitbucket, then submit a single ZIP file in the courseroom named u04a1.zip. The ZIP file should contain:

- Your updated NetBeans ProgrammableThermostat project directory.
- The screen shots of the runtime output of your program in the Console.
- The Word document containing the content from Part 2 in the Directions.

Course Resources
u04a1 User Story

Unit 5 >> The User Interface

Introduction

A graphical user interface (GUI) is a construct through which a user interacts with software. This interface uses metaphors such as icons, menus, and other graphical visual representations to display information and give the user cues on how to interact with the interface. Most often, graphical interfaces are manipulated by the user using pointing devices such as a mouse, trackball, stylus (pen), or the user's finger touching the screen.

A crucial step in the design of a software development project is wireframing which is part of the screen design process. It allows the user interface design the ability to layout screens according to how you they want the software user to process the information and interact with the screen interfaces. Wireframes are important to a number of different groups. They allow the user experience designers to plan the layout and interaction. From a user perspective, wireframes are the first opportunity to get an idea of how the software system will look and see how they will use the software. Finally, the developers use wireframes as blueprints to begin to construct the user interface.

Within Java, the graphical user interface framework has in the past been Java Swing. However, Oracle (who are the stewards of Java) has indicated that while Swing is not deprecated and Swing-based applications will continue to work, that JavaFX is the active and supported approach to building graphical user interfaces in Java. Using JavaFX Java developers now have support for complex animations and the ability to support touch devices such as touch-enabled screens. JavaFX has support for common touch gestures such as scrolling, swiping, pinching, rotating, and zooming. You will need to use JavaFX to construct the GUI for your application.

Java FX uses some terms from theater whereby a Java FX-based application has a stage which would equate to a particular window. Each of these stages has a scene attached to it. Each scene can have a number of controls (Buttons, Labels), other layouts (GridPane, TilePane), etc., embedded in it. This hierarchy of stage to scene to list of controls is called a scene graph. All the objects that can be embedded on a screen are Nodes. There are two types of Nodes: Branch nodes and Leaf nodes. A Branch node is a node that can contain other nodes, Leaf or child nodes. Because they can contain other nodes, a Branch node can be used to build complex objects and controls that are composed. A Leaf node is a node which cannot contain other nodes and are therefore at the lowest level of the Node structure.

Learning Activities

u05s1 - Studies

Readings

Use the Internet to complete the following:

- Joel on Software. (2001). User interface design for programmers. Retrieved from https://www.joelonsoftware.com/2001/10/24/user-interface-design-for-programmers/

- A timeless guide for designing user interfaces that focuses on programmers.

- UX Planet. (2016). Golden rules of user interface design. Retrieved from https://uxplanet.org/golden-rules-ofuser- interface-design-19282aeb06b
    - Heuristics that should be followed when building a user interface.
- UX Mastery. (2016). Wireframing for beginners. Retrieved from https://uxmastery.com/wireframing-for-beginners/
    - An overview of wireframes and how they are used in user interface design.
- Designmodo. (2016). Wireframing, prototyping, mockuping - What's the difference? Retrieved from https://designmodo.com/wireframing-prototyping-mockuping/
    - Discusses the differences between wireframes, prototypes, and mockups, and when to use each option.
- Oracle. (n.d.). JavaFX: Working with JavaFX UI components. Retrieved from https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/index.html
    - Tutorial describing how to use JavaFX to construct graphical user interfaces. Review Part 1 - About This Tutorial, and Part 2 - Using JavaFX UI Controls.
- Oracle. (2013). Using Scene Builder with NetBeans IDE. Retrieved from http://docs.oracle.com/javafx/scenebuilder/1/use_java_ides/sb-with-nb.htm#CIHDHEFE
    - Step by step instructions on how to create an initial Scene Builder (Java FX screen).

u05s1 - Learning Components

- Identify GUI related Java Classes.
- Study JavaFX capabilities.
- Understand principles of GUI creation using Java.

u05d1 - Model-View-Controller (MVC) Architecture

The Model-View-Controller (MVC) architecture is a common pattern that separates the user interface from the underlying application/business code.

- Describe how the MVC architecture pattern maps to Java and JavaFX.
- Discuss advantages of its adoption and give an example of how you would implement the MVC within an application.

Your post should be approximately 250 words. Support and cite your discussion with research from online materials, books, and other resources.

Response Guidelines

Respond substantially to the post of at least one of your peers. Your response should be approximately 200 words or more. Some approaches include providing an alternative viewpoint, substantiating your peer's claims with referenced material, expanding on the noted topic by citing experience, or conducting a review of the literature to further the exploration of the subject. Where possible, you should elaborate on the topic based on your professional experience and try to tie the topic to real-life expertise.

Course Resources

Undergraduate Discussion Participation Scoring Guide

u05d1 - Learning Components

- Study JavaFX capabilities.
- Understand principles of GUI creation using Java.

u05a1 - Develop the Operate Screen GUI (Sprint 3)

Overview

As the developer, you want to provide the Homeowner an easy and intuitive way to view, enter, and change the temperature of the interior of the home using the ProgrammableThermostat. Your ProgrammableThermostat will have a touchscreen with a graphical user interface to allow an owner to establish temperature settings. Like other programmable thermostats, your ProgrammableThermostat GUI will allow the Homeowner to view and set the current temperature.

In this assignment (Sprint 3), you create screens/classes/modules that that allow a user to display, set, and change the current setting for the ProgrammableThermostat application.

Preparation

Read the following in the Resources:

- u05a1 User Story.
- Operate Wireframe.

Directions

Add a Java class or classes (you may decide whether to implement the GUI using one or multiple classes) to your ProgrammableThermostat project to provide a GUI that allows a user to change the desired current interior room temperature. Your GUI should accurately represent the user story and reasonably approximate the Operate Wireframe diagram.

Make sure to:

- Create a GUI using a Java class or classes allowing the user to choose the desired room temperature.
- Create a GUI that reasonably approximates the Operate Wireframe.

Take a screen shot of your completed GUI to submit with your assignment.

Submission Requirements

You must upload your ProgrammableThermostat files to Bitbucket, then submit a single ZIP file in the courseroom named u05a1.zip. The ZIP file should contain:

- Your updated NetBeans ProgrammableThermostat project directory.
- The screen shot of your completed Operate Screen GUI.

Course Resources
u05a1 User Story
Operate Wireframe

Unit 6 >> Practical Software Development Considerations

Introduction

Developing a piece of software with a high level of quality that also meets budget and schedule constraints is the main goal of any software development team. Accomplishing this goal usually requires making trade-offs among competing aspects such as software functionality, quality, time-to-market, and the ability of the team to support and maintain the software going forward after implementation.

To move quickly to determine project requirements and reduce the complexity of the end product, the software development team can use concepts from Lean Principles such as Amplify Learning by engaging in early proof of concepts or construction iterations that enable the team: (a) not to build things people do not want or will not use, and (b) testing design ideas early as quickly and as cheaply as possible. The second Lean Principle is 'Deliver As Fast As Possible' because the sooner the software is delivered without major defects, the sooner that Product Owner and stakeholder feedback can be received and incorporated into the next iteration.

Stakeholder feedback is critical because that feedback is used to engage and explain and it can improve user satisfaction. By walking through the stakeholder feedback, the user interface designer can change a confusing user experience into a pleasant one. In this fashion, both the user interface designer and the software development team learn how the system that the Product Owner/stakeholders have envisioned works.

At the end of each sprint, the software development team should have accomplished one of two things (or both): (1) Build new functionality into the system under development, or (2) Gain new knowledge and insight on how the system should work. The feedback that you receive from the Product Owner/stakeholders may indicate a design idea that doesn't work or one which would require some rethinking. But in the end, throwing away a design idea should not be viewed as a failure but rather as gaining additional insight in how the software application should function effectively. The idea of embracing change as part of the software development process will be examined further in Unit 9.

Learning Activities

u06s1 - Studies

Readings

Use the Internet to complete the following:

- Agile Modeling. (n.d.). User Interface (UI) prototypes: An Agile introduction. Retrieved from http://agilemodeling.com/artifacts/uiPrototype.htm
- Agile Trail. (2015). Sprint review, a feedback gathering event: 17 questions and 8 techniques. Retrieved from http://agiletrail.com/2015/09/19/sprint-review-a-feedback-gathering-event-17-questions-and-8-techniques/
    - Discusses the Sprint Review meeting and effective ways to gather feedback from the user.
- Axosoft. (2017). How to use fast feedback loops. Retrieved from https://blog.axosoft.com/feedback-loops-agile-development/
    - Discusses the other feedback mechanisms within an Agile environment.
- UX Booth. (2015). User stories: A foundation for UI design. Retrieved from http://www.uxbooth.com/articles/user-stories-a-foundation-for-ui-design/
    - Focuses on user stories as the basis for user interface design.

u06s1 - Learning Components

- Understand principles of wireframing.

u06d1 - Practical Considerations for Development

Your mandate as a software developer is clear: Take the software product concept that exists as a customer's idea and breathe life into it.

Discuss which JavaFX capabilities (i.e., buttons, or other user interactive controls) are most appropriate for building the Smart Thermostat GUI in the course project.

Your post should be approximately 250 words. Support and cite your discussion with research from online materials, books, and other resources.

Response Guidelines

Respond substantially to the post of at least one of your peers. Your response should be approximately 200 words or more. Some approaches include providing an alternative viewpoint, substantiating your peer's claims with referenced material, expanding on the noted topic by citing experience, or conducting a review of the literature to further the exploration of the subject. Where possible, you should elaborate on the topic based on your professional experience and try to tie the topic to real-life expertise.

Course Resources

Undergraduate Discussion Participation Scoring Guide

u06d1 - Learning Components

- Understand principles of GUI creation using Java.

u06a1 - View Program Screen GUI (Sprint 4)

Overview

It is time to extend the GUI created in u05a1 by creating a second screen that will allow the Homeowner to set a schedule for what the temperature should be when they are awake, leave the house during the day for work, evening hours, sleeping, etc. The Homeowner should also be able to set date and time ranges for weekend hours.

In this assignment (Sprint 4), you will create screens, classes and modules that allow a user to set temperatures for various day and times throughout the week for the ProgrammableThermostat.

Preparation

Review the following in the Resources:

- u06a1 User Story.
- View Program Screen Wireframe.

Directions

Add Java classes to your ProgrammableThermostat project that create a graphical user interface that allows a user to set the future interior room temperature for a given week's day/time. Your GUI should accurately represent the user story and reasonably approximate the View Program Screen diagram.

Make sure to:

- Create a GUI that reasonably resembles the View Program Screen Wireframe.
- Create GUI functionality using Java classes that allow the user to program the date, time, and temperature on a single screen.

Include a screen capture of your completed Program Temperature GUI to submit with your assignment.

Submission Requirements

You must upload your ProgrammableThermostat files to Bitbucket, then submit a single ZIP file in the courseroom named u06a1.zip. The ZIP file should contain:

- Your updated NetBeans ProgrammableThermostat project directory.
- The screen shot of your completed Program Temperature GUI.

Course Resources

View Program Screen Wireframe

u06a1 User Story

Unit 7 >> Software Testing

Introduction

There are a number of methods of testing software, each of which examines a different aspect of the software under test.

White Box Testing

White box testing is performed when the software and its internal configuration are known. It requires programming skills to identify all paths through the software. This type of testing is very effective in detecting and resolving problems because bugs can often be found before they cause problems.

Black Box Testing

Black box testing is performed when the software and its internal configuration are not known. The tester selects valid and invalid input and determines the correct output. An advantage of this type of testing is that it is unbiased and the testing is performed from the perspective of the tester rather than the developer. A disadvantage of this type of testing is that the test cases can be difficult to design and testing every possibility is unrealistic because of time constraints.

Regression Testing

Regression testing determines if new code has changed or altered old functions. This type of testing typically occurs when new code has been created and the software no longer functions as it had previously. This is a type of quality control measure is used to ensure that new code still produces the desired results.

Software Security Testing

Software security testing validates that the security measures programmed cannot be breached. Software security is extremely important. Software developers and the software industry have had to add a level of security to their programs to correct vulnerabilities or to determine the breaches that may occur and plan accordingly.

It is helpful if the software developers understand the motivations of hackers. There are many reasons why someone would intentionally breach a computer system:

- Challenge: The hacker is bored and wants some excitement. Ultimately, he or she intends no harm. Rather, this hacker only wants a thrill.
- Curiosity: The hacker ultimately intends no harm, but is simply curious about the information that he or she can find in any given system.
- Computer Usage: The hacker has an agenda. This individual wants to use the system's owner as a decoy, perhaps to cover up his illegal and harmful activities.
- Vandalize: The hacker intends to destroy computers or alter information.
- Stealing: The hacker wants information and is going to take it without permission.

Threat Modeling

One way to help ensure that computer software has built-in security measures is by threat modeling. Threat modeling is a formal process that encompasses many phases to develop a plan. These phases are constructed so that a team of individuals can try to anticipate the breaches that may occur and determine how those breaches can be prevented. They also must plan for any vulnerabilities that cannot be corrected. Part of this phase involves trying to break the program or system so that bugs can be detected and corrected at the earliest possible stage. It is important that security measures are considered from the beginning of the software design/development phases.

Configuration testing

Configuration testing is the most common first step in testing an application. This approach takes a specific software application and uses it across different types of hardware, drivers, interfaces, and peripherals. When testing, it is important to note that you will want to test the appropriate hardware, peripherals, drivers, and other equipment with the attributes of the application. For example, if the application requires high usage of graphics and sound, then those respective hardware and peripherals should be tested along with the application.

Compatibility Testing

Compatibility testing ensures that the application can work with related or other types of software with which it was designed to work in conjunction. Cutting and pasting text from one application to another is an example of how certain applications should be and are compatible.

Usability testing

Usability testing is a method where testers must act as the user of the application. Have you ever experienced an application that was completely unfriendly to users? Usability testing eliminates, or at the very least tries to reduce, this hardship. Testers ask themselves questions such as:

- Does this application produce the required information that the user is seeking?
- Can the user easily go from one screen to another logically?

Because programming is so complex, testing is an even more critical phase in the software engineering process. It is how quality is assured. Yet because it comes later in the life cycle—perhaps when the completion deadline is already in jeopardy—it is too often rushed. Doing it well requires careful planning, and time to explore the situations in which the software must work. Testing must ensure that what is supposed to be done is done, and also what is not supposed to be done is not done. Failure points can come from the software itself, its interaction with the hardware and the telecommunications, or even in the ways the system is used.

In this unit, you are introduced to the processes developers use to test their applications, ensuring that the software functions according to design.

Learning Activities

u07s1 - Studies

Readings

Read the following in *Beginning Software Engineering*:

- Chapter 8, "Testing," pages 173–194.

Use the Internet to complete the following:

- RBCS. (n.d.). Why most unit testing is waste [PDF]. Retrieved from http://rbcs-us.com/documents/Why-Most-Unit-Testing-is-Waste.pdf
  - An article arguing that the creation of unit test cases is a poor use of a developer's time.
- RBCS. (n.d.). Segue [PDF]. Retrieved from http://rbcs-us.com/documents/Segue.pdf
  - A follow-up article by the same author expanding his thoughts on why unit test cases are a poor use of a developer's time.
- The Holy Java. (2015). Challenging myself With Coplien's Why Most Unit Testing is Waste. Retrieved from https://theholyjava.wordpress.com/2015/01/26/
  - The author objectively discusses Coplien's essay on why unit test cases are a waste.
- ThoughtWorks. (2014). Using TDD to influence design. Retrieved from https://www.thoughtworks.com/insights/blog/using-tdd-influence-design
  - Using test-driven development in API and class design.
- Guru99. (n.d.). Test-Driven Development (TDD): Learn with example. Retrieved from https://www.guru99.com/test-driven-development.html
  - Discusses the basics of test-driven development.

Optional Readings

- NetBeans. (n.d.). Writing JUnit tests in NetBeans IDE. Retrieved from https://netbeans.org/kb/docs/java/junit-intro.html
  - Provides an overview of creating unit tests within NetBeans.

u07d1 - Test-Driven Development

Test-Driven Development (TDD) is a process within the Agile development methodology domain where developers write a test for a piece of required functionality before creating any application code.

Research and review articles about TDD and how this technique differs from traditional programming styles and how it enhances Agile development. Briefly summarize two articles that you have found and give two strong reasons why using TDD is a good idea. Also describe some of the disadvantages of using TDD for developing software.

Your post should:

- Explain how TDD differs from a more traditional programming approach.
- Summarize two articles that you found on the Internet or in the Capella Library.
- Give two reasons, with explanations, why TDD is a good approach to application development and describe any disadvantages of the approach.

Your post should be approximately 250 words. Support and cite your discussion with research from online materials, books, and other resources.

Response Guidelines

Respond substantially to the post of at least one of your peers. Your response should be approximately 200 words or more. Some approaches include providing an alternative viewpoint, substantiating your peer's claims with referenced material, expanding on the noted topic by citing experience, or conducting a review of the literature to further the exploration of the subject. Where possible, you should elaborate on the topic based on your professional experience and try to tie the topic to real-life expertise.

Course Resources
Undergraduate Discussion Participation Scoring Guide
Capella University Library

u07d1 - Learning Components

- Study alternative testing frameworks.

u07a1 - Unit Testing

Overview

Unit testing is the execution of testing software against some portion of the application under test. A "unit" test does not have to be of a certain size, it can be used to test a single method or procedure. However, it usually only tests a given portion of the application. The best way to create a unit test is to write a small application, called a "unit test" using a testing framework that provides the test harness or environment for you to execute your tests.

In this assignment you create unit tests against your application classes using a testing framework.

Preparation

- Find and download a testing framework from the Internet such as JUnit, Spring, or TestNG. A resource describing JUnit testing can be found in the Studies.

Directions

Use a testing framework such as JUnit, Spring, or TestNG to create test cases to validate the software created created in u03a1. Create at least five separate unit tests that test various portions of your ProgrammableThermostat software application.

Make sure to:

- Create appropriate test cases that effectively identify errors using a testing framework.
- Write instructions that explain how to execute each of your test cases.

Take screen shots of the results from running each of your test cases for submission with this assignment.

Submission Requirements

You must upload your ProgrammableThermostat files to Bitbucket, then submit a single ZIP file in the courseroom named u07a1.zip. The ZIP file should contain:

- Your updated NetBeans ProgrammableThermostat project directory.
- Screen shots of your test cases functioning successfully.
- Instructions for how to execute the test cases.

Unit 8 >> Quality Development

Introduction

"How many times have you been convinced, absolutely convinced, that you had either created a defect-free program or had 'tested out' all of the defects only to sadly discover that yet another 'bug' was found? One wonders how we as programmers have been able to repeatedly survive this cruel onslaught to our egos." (Radice & Phillips, 1988, p. 235)

Of course, if it were just about our egos, that would not be so bad. In reality, errors in software programming and testing could cost us our jobs, could cost businesses their financial viability, or even cost someone's life!

Consider the following:

- An Ariane 5 rocket blew up seconds into its launch due to software errors, at a cost of more than $500 million (Pfleeger, A., & Atlee, J., 2010, p. 37).
- NASA's Mars Climate Orbiter burned up in the Martian atmosphere due to an issue where the underlying software controlling the orbiter's calculated the force the thrusters needed to exert in *pounds* of force. A separate piece of software took in the data assuming it was in the metric unit: *newtons* (LA Times, 1999).
- A software failure at FirstEnergy Corp. likely was a significant contributor to a blackout that crippled most of the Northeast and parts of Canada on Aug. 14, 2003 (Verton, 2003).
- Software error played a role in the crash of a Korean airliner that killed 225 people in Guam in 1997. A recent software change with a coding error prevented a warning system from activating (Pfleeger, A. & Atlee, J., 2010, p. 515).

The programming and testing stages are at the core of software engineering projects, and they can be both the most exhilarating and exasperating times in the project life cycle. The culmination of all the previous stages manifests itself in the art and expertise of the technical staff, who are charged to create the most effective and efficient code possible to meet the requirements. Testers (who may be specially trained professionals, programmers, systems analysts, or users, or any combination of these) stand at the ready to ensure that programs accurately translate input requirements into output functions.

No matter how careful and attentive a programmer might have been, the overwhelming detail and required precision make errors in coding likely. Even when they are very careful, misunderstandings may occur between the system designers and the programmers who interpret the specifications differently. Programmers often are isolated from the business end users, which may lead to further misunderstandings.

Experience has demonstrated that the programming stage is best done with group input and review, even though many programmers prefer to work in isolation or are not assigned to work in groups. While managing the programming stage of the project is challenging, using newer techniques such as Agile development can increase the success of this critical project stage.

Code reviews, both manual (peer review) and automated are the examination of the programmer's source code to identify design, logic, and coding errors that may have been overlooked in the software development process. Through this process, the knowledge of the source code is shared among the development team, code style and quality improves, and younger members of the development team are mentored.

References

Pfleeger, S. L., & Atlee, J. M. (2010). *Software engineering: Theory and practice* (4th ed.). Upper Saddle River, NJ: Prentice-Hall.

Radice, R. A., & Phillips, R. W. (1988). *Software engineering: An industrial approach*. Englewood Cliffs, NJ: Prentice-Hall.

Verton, D. (2003). Software failure cited in August blackout investigation [Electronic version]. *Computerworld.* Retrieved from http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=87400&pageNumber=1

Hotz, R. (1999). Mars Probe Lost Due to Simple Math Error. Retrieved from http://articles.latimes.com/1999/oct/01/news/mn-17288

Learning Activities

u08s1 - Studies

Readings

Use the Internet to complete the following:

- IBM developerWorks. (2011). 11 proven practices for more effective, efficient peer code review. Retrieved from https://www.ibm.com/developerworks/rational/library/11-proven-practices-for-peer-review
    - Identifies a number of practices for effective peer reviews.
- SubMain. (2016). Why automate code reviews? Retrieved from https://blog.submain.com/why-automate-code-reviews/
    - Identifies advantages to software development teams by employing automated code review tools.
- Continuous Dev. (2015). Checkstyle vs PMD vs Findbugs. Retrieved from http://continuousdev.com/2015/08/checkstyle-vs-pmd-vs-findbugs/
    - Overview of some of the leading automated code review plugins for Java IDEs.
- Envato Tuts+. (2017). Ensure high-quality Android code with static analysis tools. Retrieved from https://code.tutsplus.com/tutorials/ensure-high-quality-android-code-with-static-analysis-tools--cms-28787
    - Advocates why using static code analysis tools improves code quality.

u08s1 - Learning Components

- Review Eclipse, IDEA, and NetBeans platforms.
- Review examples of a platform selection methodologies.
- Understand business requirements or restrictions for a development platform.
- Understand Platform requirements for a project.
- Understand how to different IDEs function against criteria.

u08d1 - Peer Reviews

As stated in the unit introduction, many developer groups look at code reviews as a legacy of waterfall development methodologies. In today's Agile environments, software development groups may put more emphasis on peer reviews where the focus is on examining the design and functionality of the software under review.

Discuss how this changed focus has improved the quality of software over the code reviews of old and what purpose those peer reviews are serving today.

Your post should be approximately 250 words. Support and cite your discussion with research from online materials, books, and other resources.

Response Guidelines

Respond substantially to the post of at least one of your peers. Your response should be approximately 200 words or more. Some approaches include providing an alternative viewpoint, substantiating your peer's claims with referenced material, expanding on the noted topic by citing experience, or conducting a review of the literature to further the exploration of the subject. Where possible, you should elaborate on the topic based on your professional experience and try to tie the topic to real-life expertise.

Course Resources

Undergraduate Discussion Participation Scoring Guide

u08d1 - Learning Components

- Study principles of code review.
- Understand the role of code reviews in software development.
- Review principles of the Agile methodology.

u08a1 - Merits of Code Reviews

Overview

A number of software development groups include code reviews as part of their verification and validation process as the developer completes the coding of a particular module or piece of software. However, other groups reject code reviews as a legacy of the waterfall development methodologies used in previous decades. Therefore, frequently it is necessary to advocate for code reviews and develop use cases for driving code reviews within the software development process.

For this assignment, imagine that you have been hired as the lead technical developer for a software development team and it is your job present the pros and cons of conducting a code review for the course project (or a similar project).

Preparation

Review the Kick-off Meeting Notes found in the Resources as needed.

Directions

Create convincing arguments both for and against using code reviews in the software development process.

Make sure to:

- Create convincing arguments *for* conducting code reviews.
- Create convincing arguments *against* conducting code reviews.
- Explain how a code review might work in an Agile environment. (Consider both manual and automatic approaches.)

Additional Requirements

- Font: Times New Roman 12 point.
- Length: 3-4 double-spaced pages not including cover page and current APA compliant references.

Course Resources

Kick-off Meeting Notes

Unit 9 >> Change Happens

Introduction

In 2005, the Swiss insurance company Zurich came up with the sensationally-spelled slogan, "Because Change Happenz." According to their website, the phrase reflected the fact that "change is an unavoidable reality." This is one of the tenets that Agile development methodologies are based on: Change will happen, so how do we as a software development team deal with it?

Change in software projects is expensive: it leads to wasteful rework and therefore it is a risk that we want to mitigate to the best of our ability. There are a number of ways to mitigate that risk. Two that immediately come to mind are reducing the likelihood of the event occurring and reducing the impact of the event.

The traditional software development life cycle (SDLC) or waterfall methodology takes the approach of investing significant time in upfront planning, requirements gathering, and design to avoid the possibility of the change adversely affecting the project. Unfortunately, over five decades of software development experience have shown that change cannot be avoided. The reason is that the changes that normally impact a software development project cannot be eliminated by significant upfront planning and analysis – the environment changes and business requirements change.

There are two main sources of change that frequently occur in software projects:

(1) The customer or stakeholders begins to see how the software is taking shape and through interactions with the software decide that the software needs to work differently.

OR

(2) An outside business change has occurred that requires the project to change. This could be integrating with some other piece of software, a government regulation needs to be supported, or there is a need to add functionality to the software to support some new business opportunity.

Neither of these situations could have been eliminated by better planning. When given an unpreventable risk the logical approach is to identify ways to reduce the impact. This is the premise that the people who came up with the Agile Manifesto are coming from. If change is going to happen, we know it is going to happen, and if it's expensive, let's work to make it cheaper.

So, how do you do that?

Only plan the details for the next few days or weeks (this is why sprints are normally two weeks in length).

Build just enough software to solve the problems that you have identified rather than trying to build software for requirements that were defined months ago.

Pay attention to software quality from the beginning of the construction process and continuously throughout that process (e.g., good design, automated regression tests, etc.) so that code is easy and safe to change.

Defer design and development decisions as long as you responsibly can so that you have time to gather more information. There is a phrase YAGNI, or "You Ain't Gonna Need It" which emerged from the Extreme Programming people that states: "Always implement things when you actually need them, never when you just foresee that you may need them."

Learning Activities

u09s1 - Studies

Readings

Read the following in *Beginning Software Engineering*:

- Chapter 11, "Maintenance," pages 241–257.

Complete the following on the Internet:

- Agile Modeling. (n.d.) Agile requirements change management. Retrieved from http://agilemodeling.com/essays/changeManagement.htm
  - Explains a process incorporating change management within Agile.
- ThoughtWorks. (2013). Providing 'just enough design' can make Agile software delivery more successful. Retrieved from https://www.thoughtworks.com/insights/blog/providing-just-enough-design-can-make-agile-software-delivery-more-successful
  - How using minimal design aids in development flexibility in the face of change
- Prosci. (n.d.). Adapting and adjusting change management in Agile. Retrieved from http://blog.prosci.com/adapting-and-adjusting-change-management-in-agile
  - Another perspective on incorporating change management within Agile.

Capella Documents

Read the following documents related to your course project.

- u09a1 User Story.
- User Story Diagram Change 1.
- User Story Diagram Change 2.

u09d1 - Change Management

Research change management for Agile and Waterfall on the Internet. Discuss the following:

- How does change management within the Agile process differ from that of traditional software development?
- What are the implications for software construction?

Your post should be approximately 250 words. Support and cite your discussion with research from online materials, books, and other resources.

Response Guidelines

Respond substantially to the post of at least one of your peers. Your response should be approximately 200 words or more. Some approaches include providing an alternative viewpoint, substantiating your peer's claims with referenced material, expanding on the noted topic by citing experience, or conducting a review of the literature to further the exploration of the subject. Where possible, you should elaborate on the topic based on your professional experience and try to tie the topic to real-life expertise.

Course Resources

Undergraduate Discussion Participation Scoring Guide

u09d1 - Learning Components

- Understand how to adapt to change requests.

u09a1 - Revised User Story

Overview

At the end of every sprint, the respective business unit and stakeholders will participate in a Sprint Review Meeting in which the current software is demonstrated to the Product Owner, sponsors, customers, and users. The feedback from all the stakeholders may cause a particular User Story to be changed and reintroduced in a future sprint.

Scenario

The u06a1 User Story has not been determined to be "Done" by the Product Owner. A change has been approved by the Product Owner and the Project Manager and is now encapsulated in the u09a1 User Story.

Preparation

Review the following items found in the Resources:

- u09a1 User Story.
- User Story Diagram Change 1.
- User Story Diagram Change 2.

Directions

Add or modify Java classes in your ProgrammableThermostat project to support the changes documented in u09a1 User Story and the change diagrams.

Make sure to do the following:

- Modify your ProgrammableThermostat Java classes to reflect new user requirements for setting temperatures.
- Modify your ProgrammableThermostat Java classes to reflect new user requirements for setting times.

Submit screen shots of your GUI screens that document that your updates work.

Submission Requirements

You must upload your ProgrammableThermostat files to Bitbucket, then submit a single ZIP file in the courseroom named u09a1.zip. The ZIP file should contain:

- Your updated NetBeans ProgrammableThermostat project directory.
- The screen shots of your updated ProgrammableThermostat GUI.

Course Resources
u09a1 User Story
User Story Diagram Change 1
User Story Diagram Change 2

Unit 10 >> Deployment and Documentation

Introduction

Some Agile developers maintain that their products are self-documenting. However, it is important to learn good documentation skills, even with improved methods. User and operator training and good documentation are critical to success in a development project. Good user training and documentation are critical because:

- Unless users are adequately trained, they may become frustrated in their efforts to use a new system. They also may not be able to use a system effectively and efficiently. The result is that despite the quality of the development effort and the product, users might develop a negative impression and the perception of the system will be that it is less than successful.
- Documentation is needed by the operators—the technical staff who ensure that processes are executed on schedule, that disk storage is adequate, that databases are backed up, and so forth.
- Documentation is needed by maintenance programmers when a problem arises or some enhancement must be made. The original developers will move on to other projects or other companies and be unavailable to provide answers. To extend the life span of systems and to preserve them as corporate assets, systems must be updated as business needs change.
- Documentation is needed to support disaster recovery.

Learning Activities

u10s1 - Studies

Readings

Read the following in *Beginning Software Engineering*:

- Chapter 9, "Deployment," pages 203–214.

Use the Internet to complete the following:

- Chron. (n.d.). How do I write a standard operations procedures manual? Retrieved from http://smallbusiness.chron.com/write-standard-operations-procedures-manual-2596.html
  - Guide for creating an operations procedures manual.
- WikiHow. (n.d.). How to create a user manual. Retrieved from https://www.wikihow.com/Create-a-User-Manual
  - Guide for creating a user manual.

u10d1 - Acceptance and Delivery

Discuss the different requirements when creating operating instructions, user guides, and manuals and reference documentation for maintenance.

What information would likely be required by a technical writer to create one of these documents?

Your post should be approximately 250 words. Support and cite your discussion with research from online materials, books, and other resources.

Response Guidelines

Respond substantially to the post of at least one of your peers. Your response should be approximately 200 words or more. Some approaches include providing an alternative viewpoint, substantiating your peer's claims with referenced material, expanding on the noted topic by citing experience, or conducting a review of the literature to further the exploration of the subject. Where possible, you should elaborate on the topic based on your professional experience and try to tie the topic to real-life expertise.

Course Resources

Undergraduate Discussion Participation Scoring Guide

u10d1 - Learning Components

- Understand user manual requirements.

u10a1 - Thermostat Operating Instructions

Overview

Quite often, developers are tasked with creating the software user guide or user directions as part of the wrap-up tasks for any significant software application. In this effort, remember that you need to address what the end user cares about, which is how does the application supports what they want to accomplish. Identify tasks that a user would want to complete such as "program the thermostat for recurring weekends" or "program the thermostat for a holiday" using step-by-step instructions with screenshots of your application interface as examples.

In this assignment you draft basic user operation directions for the thermostat that will be handed off to a technical writer for further refinement.

Directions

Create a Word document with operating instructions for the thermostat that are appropriate for hand-off to a technical writer. Include all defined functionality of the Thermostat using screenshots of your user interface as supporting material.

Submission Instructions

- Font: Times New Roman, 12 point.
- Double-spaced lines.

u10d2 - Course Reflection

Reflect upon your overall course experience. Include aspects of the course that you found most beneficial and other aspects in which you feel can be improved. What did you find to be most challenging in this course?

Response Guidelines

Comment on the post of at least one other learner.

Course Resources

Undergraduate Discussion Participation Scoring Guide