***Lab 3 – Maintenance***

Date Assigned: Thursday, August 29

Date due: **Thursday August 29**

**Learning Objectives**

Upon successful completion of this lab exercise, the student will be able to:

* Review/Understand what Maintenance Projects are about; and
* Review/Understand refactoring.

**Instructions:**

1. Download a copy of the Lab 3 file from the Moodle page.
2. Save this document as a Word document named YourUserName\_K30\_L03\_Maintenance.docx. The document will hold your answers for your lab.
3. Research from class notes and resources, and online resources to answer the questions.
4. Write your answers to the lab questions in the appropriate locations in this file and be sure to save.
5. When you are ready to have your lab marked, notify the professor.
6. When you are finished, submit to Moodle.

To do:

**Use our class notes, discussions, and online resources to answer the following questions.**

1. For each of the following, include 2 or 3 sentences to explain the type of maintenance and include an example. Make sure you put it in your own words, i.e. don’t just cut and paste.
   1. What is corrective maintenance? Give an example.

Corrective maintenance is when there are errors in the system that cause it to not function as intended. The errors that induce corrective maintenance typically include logic errors, implantation errors, or a general misunderstanding of the requirements. A button causing the page to crash is an example of corrective maintenance.

* 1. What is perfective maintenance? Give an example.

Perfective maintenance is when the system works as intended but could be improved. When there are recommendations on streamlining a part of the system or improve efficiency, perfective maintenance takes place. An example would be turning a form that works but reloads the page many times when going through it into one that does not cause page reloads.

* 1. What is adaptive maintenance? Give an example.

Adaptive maintenance happens when technologies or ideas that are used in the system become outdated. These could include the hardware platform, the OS platform, business rule changes, legal changes. If Bill 17 requires the Heritage Nursing Application to also be offered in French when it doesn’t already, this would be adaptive maintenance.

* 1. What is preventative maintenance? Give an example.

Preventative maintenance is done to prevent bugs before they happen. If there is knowledge that a change that is out of our control will affect the system, preventative maintenance could help mitigate the number of issues that occur when the change occurs. For example, if there is knowledge that .NET 6 will become outdated next year, preventative maintenance could allow us to prepare for this switch, so the system doesn’t crash when the framework becomes outdated.

1. Name and explain 2 challenges associated with each of the types of maintenance.
   1. Corrective maintenance:
      1. Finding the root cause
      2. Accidentally introducing new problems
   2. Perfective maintenance:
      1. The architecture of the system may not scale with perfective maintenance changes
      2. Each enhancement evolves the system into one that is very large and hard to maintain
   3. Adaptive maintenance:
      1. User may not see a change in the system, so they may not value it
      2. The design or architecture of the system may not adapt.
   4. Preventative maintenance:
      1. Hard to get approved with limited resources
      2. Requires careful regression testing to ensure it is an acceptable new baseline
2. What is refactoring? Give 3 examples of changes that can be considered refactoring.

Refactoring is a form of preventative maintenance where a program’s external behaviour remains the same, but the source code and internal structure is improved. Sometimes in development code is written quickly and careless with the intention of refactoring later due to time or resource constraints. Some examples of changes that could be considered refactoring are removing duplication, reducing coupling between the systems parts, and making code less complex and more understandable so it is easier to maintain in the future.

1. Explain why testing is so important for successful refactoring. How would the absence of existing test assets affect the effort required?

Testing is important for successful refactoring because being that the system is function at this point, it is important to make sure that any change made does not affect the functionality of the system in an unexpected way. The absence of existing test assets would make it hard to confirm that all parts of the system remain functional after refactoring, and it would cost extra resources to make these test assets during refactoring.

1. Explain the term technical debt. How can it lead to the need to refactor code?

Technical debt is a measurement of how much additional work is needed because of having chosen an easier route earlier in development. This is often done on purpose when under limited resources with the intent of refactoring later. This can limit the system and delay development when a roadblock is encountered that could have been prevented but was dumped into technical debt instead as an effect of quicker development.

1. Refer to the following link: [Software maintenance - Wikipedia](https://en.wikipedia.org/wiki/Software_maintenance). Explain Maintenance debt. How does obsolescence factor into maintenance debt? Is it possible for Maintenance debt to make it impossible to support a product?

Maintenance debt is the accumulation of unresolved issues, or outdated practices that happens when maintenance tasks are postponed or neglected. Obsolescence happens when components, tools or practices become outdated or no longer supported. This can cause increased costs, security risks, an incompatibility. Maintenance debt can make it impossible to maintain a system because of high costs, critical failures, security vulnerabilities, and loss of support.

1. Use the website <http://jamesshore.com/Agile-Book/refactoring.html> to answer the following questions.
   1. In a couple of sentences describe, in your own words, *reflective design*.

An approach that allows a system’s structure and behaviour to be introspected and modified at runtime. This enables the system to adapt to new requirements dynamically without recompilation or downtime.

* 1. Describe 4 examples of *code smells* that the author describes.

Primitive Obsession -> High level concepts with primitive designs, like using a decimal to represent dollars.

Data Clumps -> Several primitives represent a concept as a group. For example, sever strings represent an address, instead of it having its own class.

Wannabee Static Class -> Class that contains methods but no meaningful per-object state.

Coddling Nulls -> Errors hidden by null values that find their way deeper into the application instead of addressing the root cause.

1. Read [Coding conventions - Wikipedia](https://en.wikipedia.org/wiki/Coding_conventions) and answer the following questions.
2. What are Code Conventions?

A set of guidelines for a programming language that recommend a programming style, practices, and methods for each aspect of a program written in that language.

1. Why are they important for software maintenance?

They are important for software maintenance because the closer you follow these guidelines the more universal your code will be, thus making it more understandable and easier to maintain.

1. What does LDAP stand for? How is it used?

Lightweight Directory Access Protocol -> It is used to access and maintain directory information services over a network. These directory services could be resources like files, printers, users, devices, and servers. For example, if an organization stores information for all their printers in a directory, an LDAP server can enable users to search for a printer, locate it, and connect to it.

1. Explain what an IIS AppPool is and describe 2 benefits.

An Internet Information Services application pool is a collection of URLs that is routed to one or more worker processes. They can be used to:

1. Isolate one or more applications into their own process.
2. You can run run applications under specific frameworks

Mark Breakdown:

|  |  |
| --- | --- |
| **Question** | **Mark** |
| 1. Types of Maintenance | 8 |
| 1. Challenges for each | 8 |
| 1. Refactoring, examples | 4 |
| 1. Testing and refactoring | 4 |
| 1. Technical debt | 4 |
| 1. Maintenance debt | 4 |
| 1. Reflective design and code smells | 6 |
| 1. Coding conventions | 4 |
| 1. LDAP | 2 |
| 10. IIS AppPool | 2 |
| Organization, English, handed in properly | 1 |
| Total | 47 |

**To submit**

When you have completed the lab save the file and upload the following to Moodle: YourUserName\_K30\_L03\_Maintenance.docx.