Jin L.C. Guo

COMP 303 Software Design
Introduction

Why are we here?

Design Patterns: Ele… 1994 | Domain-driven design Eric J. Evan… | Code Complete Steve McCo… | Head First Design Patte… 2004 | Refactoring 1999 | Clean Architecture:… Robert Cecil… | Agile Principles, P… 2002 | Code: The Hidden Lang… 1999 | A Philosophy of Software … John Ouster… | Building Microservice… Sam Newma… | The Algorithm Design Manual Steven Skie…

Designing Data-Intensi… Martin Klepp… | Pattern-oriented Soft… 1995 | Refactoring to Patterns Joshua Keri… | Beautiful Code 2007 | Site Reliability Engineering:… 2016 | Software Systems Arc… 2005 | Applying UML and patterns Craig Larma… | Don't Make Me Think Steve Krug, … | Head First Object-Orien… Brett McLau… | Design patterns expl… Alan Shallo… | The Design of Design Fred Brooks,…

… …

Software Design is about?

- Write five things/activities related to software design.
- Write your motivation of being here.

Software Design is about?

Managing COMPLEXITY

Image source: https://sourcemaking.com/files/sm/images/spagett.jpg

Software Design is about?

Complexity leads to

change amplification,

cognitive load, and

unknown unknown

Software Design is about?

Coping with imperfect world --

the potential defects from

- *Client*
- *Developer/User*
- *Environment*
- *Software*

NASA: "Astronauts would not make any mistakes. They were trained to be perfect."


“When everybody actually is *out to get you, paranoia is just good thinking”*

Software Design is about?

Communication with

Computers

PEOPLE

"… the ratio of time spent reading versus writing is well over 10 to 1" – Robert C. Martin

Definition of Software Design

(As a process) the construction of abstractions of data and computation and the organization of these abstraction into a working software application.

- Abstractions – variables, classes, objects, etc.
- Organization – modularized in a flexible and maintainable manner
- Working – correctly functioning (specification, testing)

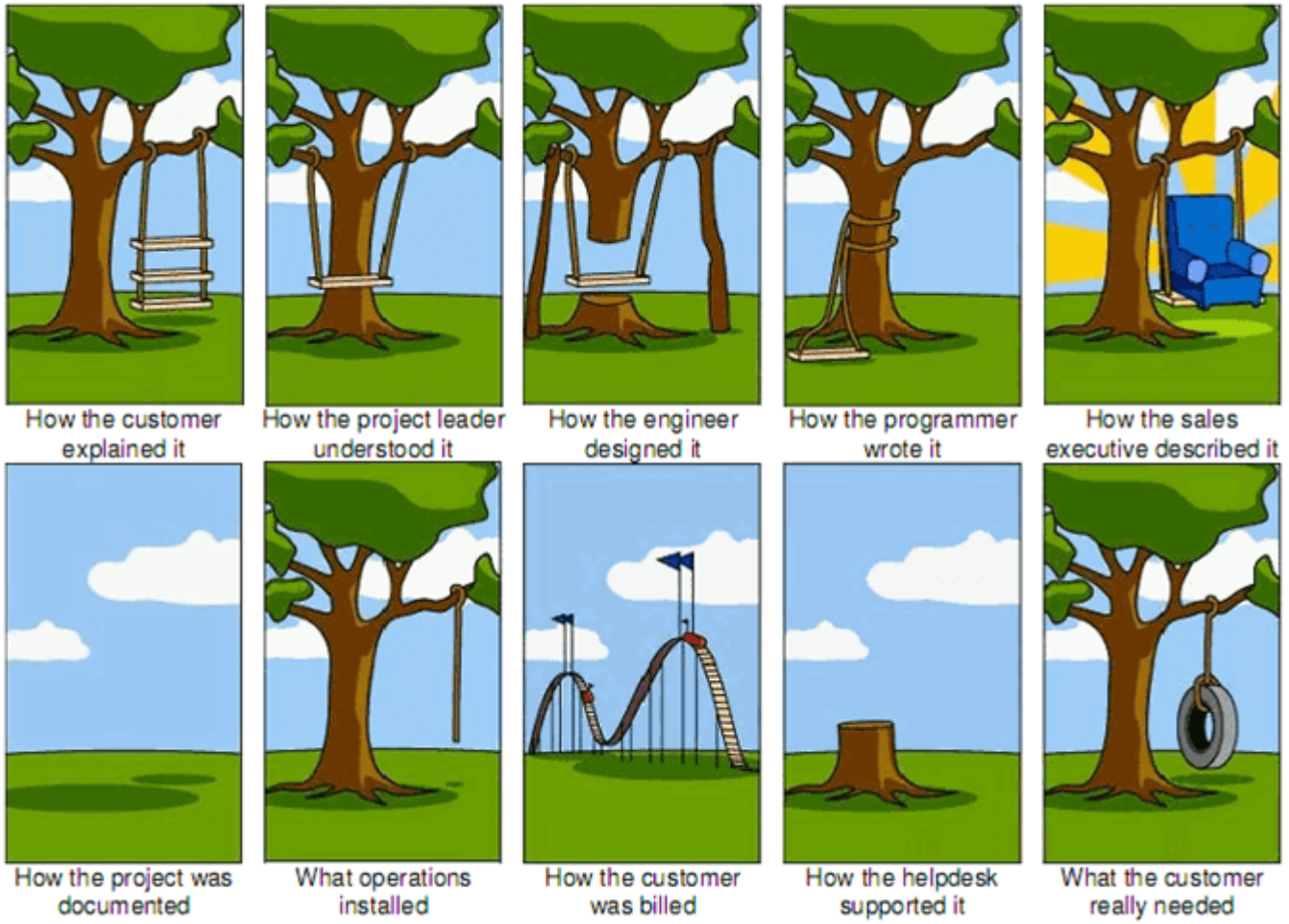
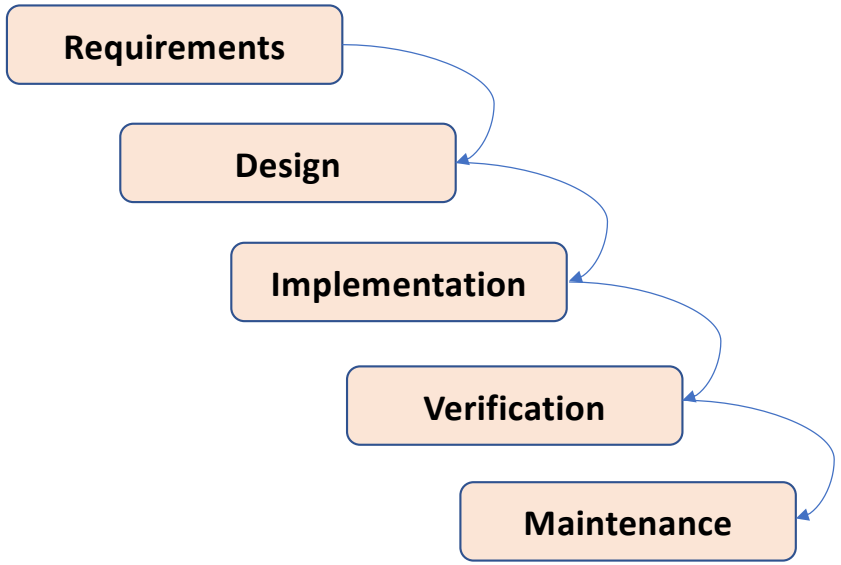


Image Source: http://tamingdata.com/wp-content/uploads/2010/07/tree-swing-project-management-large.png

Design in Software Engineering

Software development process



Design in Software Engineering

Software development process

Requirements

Elicit *Analyze* *Document*

Image Source: https://www.outsystems.com/blog/truth-about-non-functional-requirements.html

Design in Software Engineering

Software development process

Conceptual Design

Technical Design

*Recognize the **components**, **connections**, and **responsibility** of the software product.*

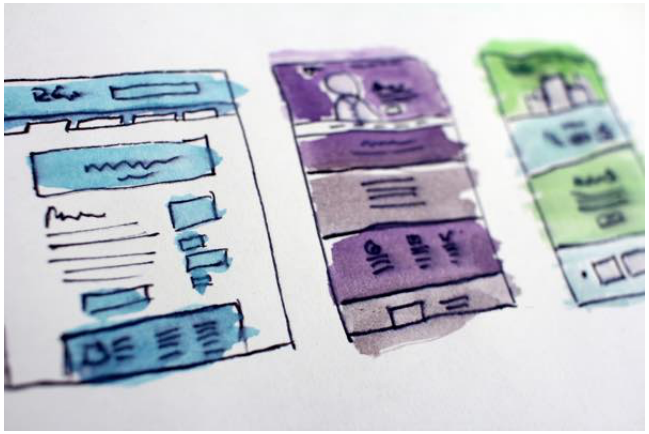


Image source: http://maryshaw.net/wp-content/uploads/ui-design-course.jpg

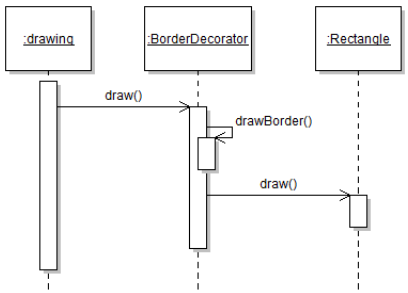
Design in Software Engineering

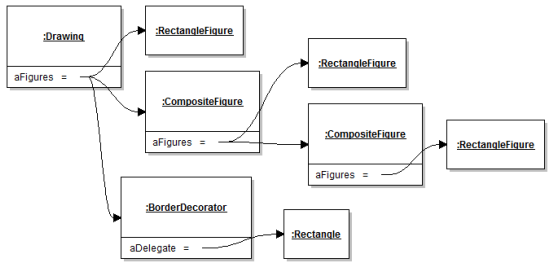
Software development process

Conceptual Design

Technical Design

describe how the responsibilities are met.



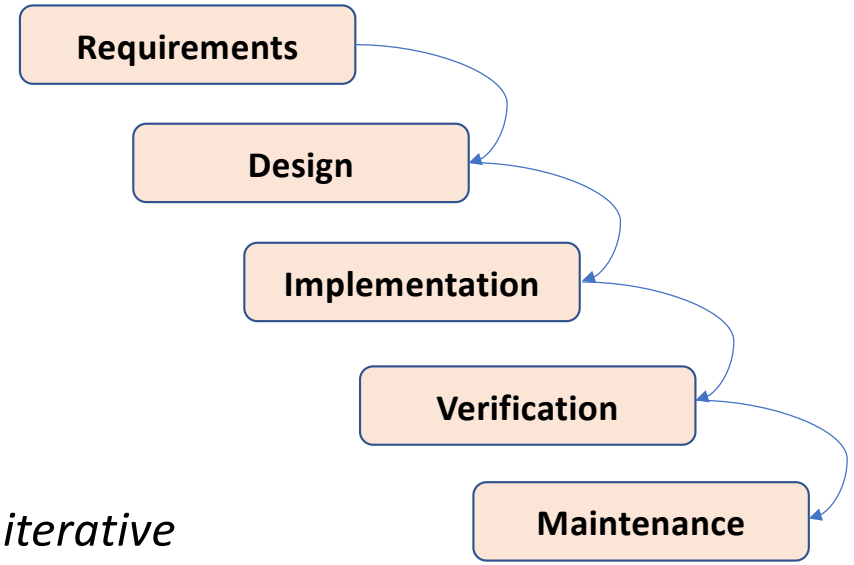


Design in Software Engineering

Software development process

Waterfall

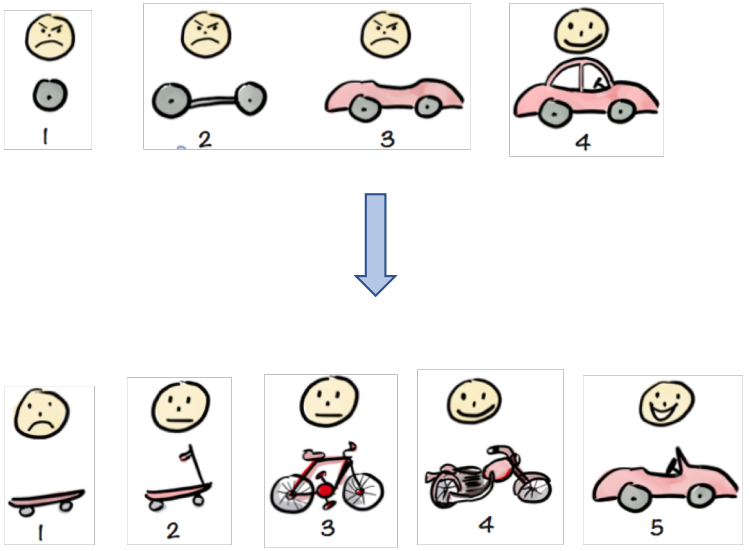
Agile



Emphasize incremental and iterative development and delivery

Agile Practices

Waterfall



Agile

Emphasize incremental and iterative development and delivery

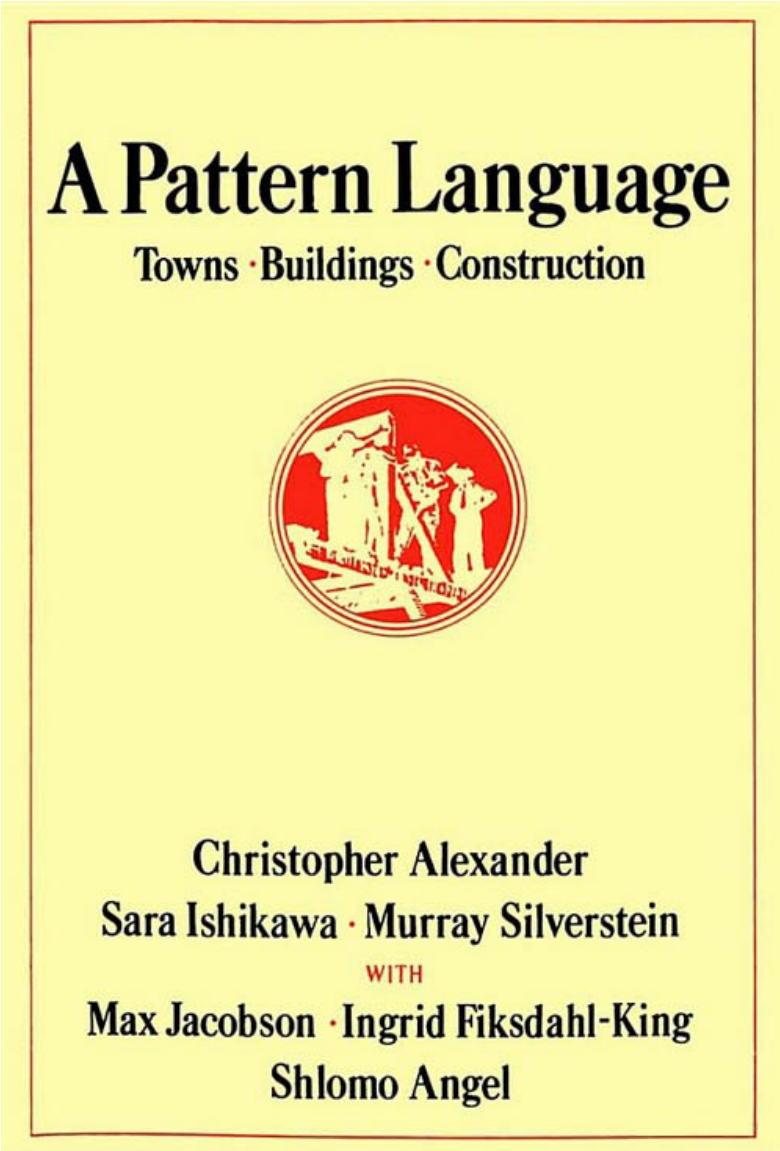
Image source: https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp

Agile Practices

Image source: https://www.agilealliance.org/agile101/subway-map-to-agile-practices/

Storing and Sharing Design knowledge

- Interval -- Human head
- External
  - Source code, including identifier, comments
  - Design documents, including diagrams, blog post
  - Discussion platforms and version control system
  - Design Patterns: name, problem, solution, consequences



Storing and Sharing Design knowledge

- *"The elements of this language are entities called patterns. Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."* — Christopher Alexander

- How to achieve greater beauty and livability (Wholeness) – within the built environment

Storing and Sharing Design knowledge

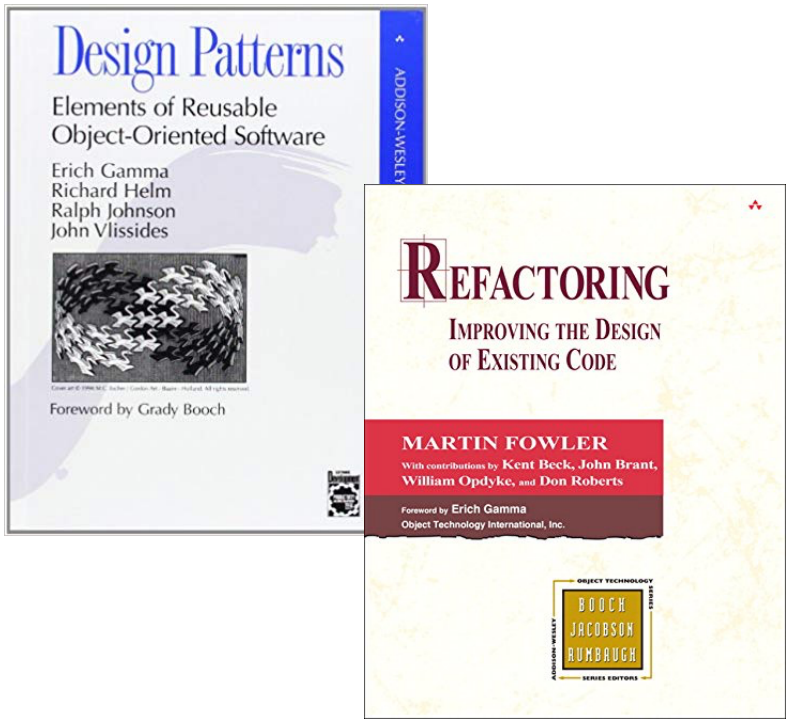
- Design pattern
  - Name
  - Problem
  - Solution -> Solution template
  - Consequences

Storing and Sharing Design knowledge

- Design pattern
  - Name
  - Problem
  - Solution -> Solution template
  - Consequences

- Code smell or Antipatterns
  - God classes
  - Primitive Obsession

  ……



This course is about?

- Design Principles (separation of concerns, encapsulation, substitutability, interface segregation, etc.)
- Design patterns;
- Design techniques such as UML Diagrams and Design by Contract;

Properly *Explain* and *Apply*

This course is about?

- Programming language mechanisms such as exception handling, concurrency and synchronization, reflection;

Effectively *Use*

This course is about?

- The quality of design solutions *Analyze* and *Evaluate*
- Design smells *Identify*
- Refactoring techniques *Properly apply*

Background Knowledge

- You have taken COMP 206 and COM 250;
- You are able to
  - Understand and use basic data structures (such as arrays, hash tables, trees and lists);
  - Understand the basic notions of object-oriented programming (such as objects, references, self, interfaces, and subclassing);
  - Write Java programs to solve small and well-defined problems (given the specification);
  - Use a revision control system to organize your work;
  - Use a debugger to trace through execution and inspect run-time values.

Logistics

- Syllabus:
  http://jguo-web.com/COMP303_Winter2019/
- Discussion Forum:
  http://piazza.com/mcgill.ca/winter2019/comp303

Acknowledgement

- Course designed and evolved by:
  - Prof. Martin Robillard.

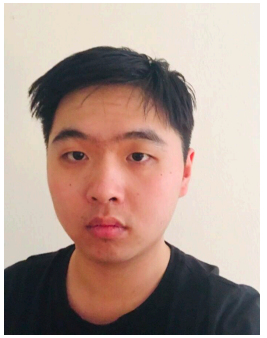
- TA Team

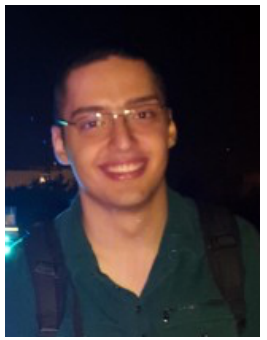

Mathieu Nassif


Deeksha Arya


Alexander Nicholson


Cheryl Wang

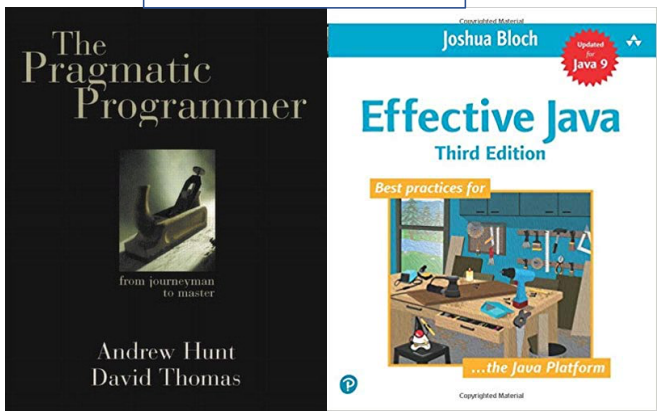

Shi Yan Du


Kian Ahrabian

Reference Material

- *COMP 303 Lecture note* by Martin Robillard (LN)
- *The Pragmatic Programmer* by Andrew Hunt and David Thomas, Addison-Wesley, 2000. (PP)
- *Effective Java* by Joshua Bloch, 2nd ed., Addison-Wesley, 2008; or 3rd ed. 2018. (EJ)

Introduction to Software Design with Java



Assessment and Evaluation

- Class and online Participation
  - Participation sheets
  - Extra points from online participation (top 10% contributors)
- Lab Tests
  - Four sessions
  - Book your time with TAs
- Mid term
  - Feb 28th 6pm
- Final

Ethics & Integrity

- ACM Code of Ethics and Professional Conduct

https://ethics.acm.org

- Academic Integrity

https://www.mcgill.ca/students/srr/academicrights/integrity

Tentative Schedule

Week	Tuesday Class	Thursday Class
1	Introduction	Encapsulation
2	Encapsulation	Types and Polymorphism
3	Types and Polymorphism	Types and Polymorphism
4	Object State	Object State
5	Unit Testing (taught by TAs)	Unit Testing (taught by TAs)
6	Composition	Composition
7	Inversion of Control	Inversion of Control
8	Review	**Midterm (Feb 28th)**
9	**Study Break**	**Study Break**
10	Inheritance	Inheritance
11	Design Patterns	Design Patterns
12	Concurrency	Concurrency
13	Refactoring	Refactoring
14	More Topics in Software Design	More Topics in Software Design

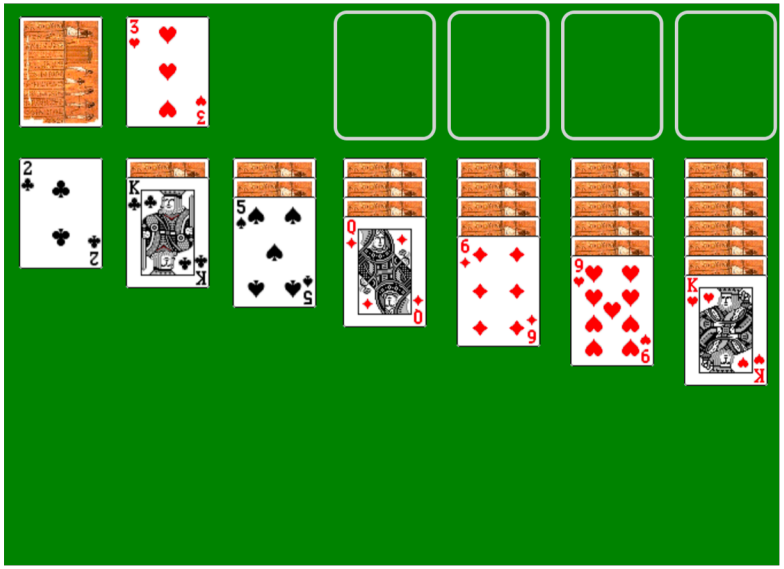
To-do list

- Sign up the Discussion Forum:
  http://piazza.com/mcgill.ca/winter2019/comp303
- Reading:
  - ACM Code of Ethics and Professional Conduct
  - LN:M0, PP:1, 6
- Assignment:
  - Setup your environment with JDK 8 and Junit 4
  - Exercise:

  https://github.com/jin-guo/COMP303_Winter2019/blob/master/Assignments/M0-Exercise.md

In-Class Activity if time

- Imagine that you are designing the Solitaire game. What are the requirements?
- How will you sketch the conceptual design of the system? Sketch the major components, responsibilities, and their collaborators.



Class Name	
Responsibilities	Collaborators

Write down your questions