Software Development Principles and Practices

Requirements and Specifications

Week 4

Prejudice disabled me from falling in love with others,
and pride shuns others away from me.

— Jane Austen, Pride and Prejudice

Objectives

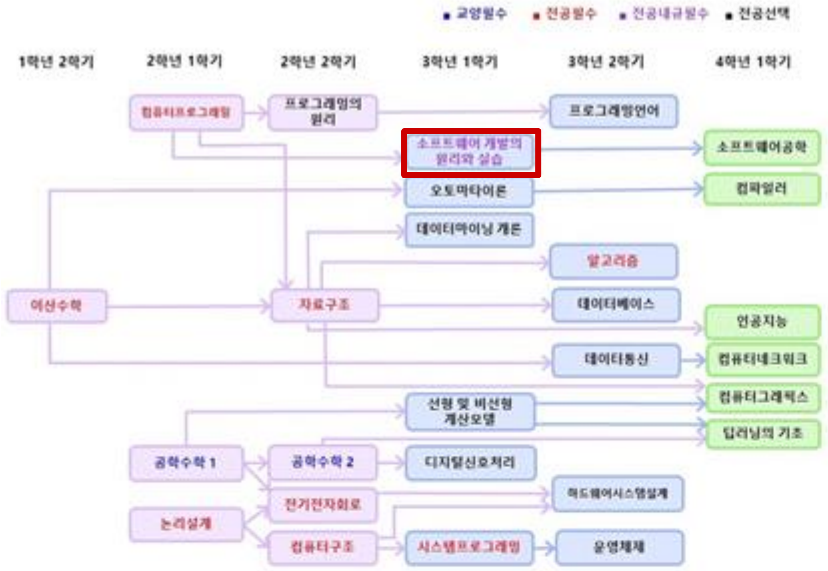
- Learn how to capture user requirements
- Learn how to turn requirements into specifications

Contents

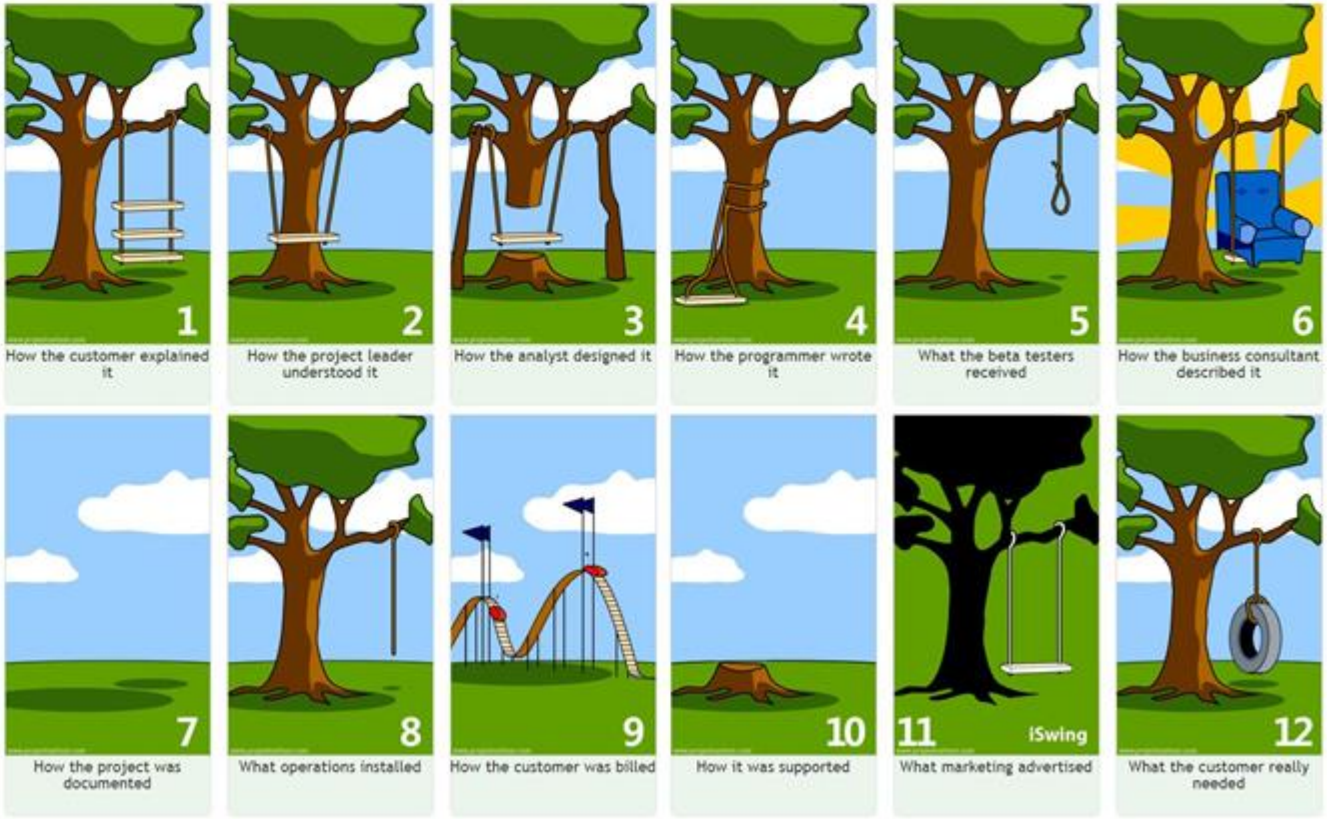
- Requirements
    - User stories
    - Wireframe
- Specifications
    - Type of specifications
    - Class diagram

What to Program vs. How to Program

- Most prior courses deal with *“How to program?”*
  - Computer Programming, Data Structures, Algorithms, etc
- This lecture mainly focuses on *“What to program?”*



Difficulty of Capturing Requirements


1
How the customer explained it
2
How the project leader understood it
3
How the analyst designed it
4
How the programmer wrote it
5
What the beta testers received
6
How the business consultant described it
7
How the project was documented
8
What operations installed
9
How the customer was billed
10
How it was supported
11
iSwing
What marketing advertised
12
What the customer really needed

Source: Project Management - A Tree Swing Story

Importance of Requirements

- Many projects were cancelled or challenged due to [1]
  - Lack of user involvement
  - Incomplete user requirements and specifications
  - Changing user requirements and specifications

[1] The CHAOS Report (1994)

Requirements Specifications

- Functional
  - Input-output behaviors
- Non-functional
  - Performance, privacy, usability, reliability, etc.
- Constraints
  - Hardware resource, testing infrastructure, etc.

Requirements Engineering Process

- Eliciting
  - Gathering and helping users to define requirements
- Analyzing
  - Formularizing the requirements gathered
- Documenting
  - Formal or semiformal documentation

Eliciting Requirements (1/2)

- Goals
  - Gathering requirements of stakeholders
- Strategies
  - Interviews
  - Observations
  - Workshops
  - Prototyping

Source: What is Elicitation: Top Requirement Elicitation Techniques for 2023

Eliciting Requirements (2/2)

- Interview[1]
  - Interview with stakeholders to build trust and collaboration
  - Structured, unstructured, and semi-structured interviews
  - Can be biased by interviewer and interviewee
- Prototyping[2]
  - Building a model to demonstrate, evaluate, and test
  - Low-fidelity and high-fidelity prototypes
  - Visual and tangible way for communication and collaboration
  - High demand on time and resource
- Studied in Human-Computer Interaction (HCI) domain

[1] What are the benefits and challenges of using interviews as a requirements elicitation tool?

[2] How do you use prototyping as a requirements elicitation tool to get feedback from users?

Challenges in Eliciting Requirements

- Problems of scope
  - Some user requirements may be out of scope
  - Every user wants different functions
- Problems of understanding
  - Users do not know well about what they need
  - User requirements may be ambiguous or untestable
- Problems of volatility
  - Requirements change over time

Analyzing Requirements

- Formularizing the requirements into actionable items
- Techniques
  - User stories
  - Software Requirements Specification (SRS)

User Stories (1/2)

- Description of a system from the perspective of an user
  - Small scale and easy to use
  - Explain purpose and the value of the feature
- Focus on who, what, and why
- Capture functional requirements
- A well formed user story will avoid unnecessary discussion, reduce rework and shorten development time

User Stories (2/2)

- Connextra template
  - As a <role>, I can <capability>, so that <receive benefit>
  - As a <role>, I want to <capability>, so that <receive benefit>
- Examples
  - As a privacy enthusiast, I can encrypt my video chats, so that my data is not leaked even if the app servers are hacked.
  - As a free plan user of the cloud service, I can indicate folders not to back up, so that my cloud storage isn’t filled up with things I don’t want to save.

Creating User Stories (1/3)

- Bad user story example
  - `As a user, I want to login.`
    - What does “login” mean?
    - What does “user” mean?

Source: Perfecting the Login User Story

Creating User Stories (2/3)

- Think about success scenario
  - Enter a username and password, then click submit
  - The system searches the username
  - If found, the system compares encrypted password
  - If match, the latest login date is updated
  - The system lookups the group of the user
  - Screen displays the results

Source: Perfecting the Login User Story

Creating User Stories (3/3)

- Improved user story example
  - `As a registered user, I can authenticate against my profile and retrieve my group membership, so that I can access my data and features permitted to the group.`

Source: [Perfecting the Login User Story]

User Story Evaluation

- Good user stories should be … ***INVEST***[1]!
  - **I**ndependent
    - Should not depend on others
  - **N**egotiable
    - A story is not a contract, but part of a conversation
  - **V**aluable
    - A story should be valuable to someone
  - **E**stimable
    - Development cost must be estimable
  - **S**mall
    - User stories should be small and cover single thing
  - **T**estable
    - Developers should be able to set acceptance criteria

[1] Wake, Bill. *"INVEST in good stories, and SMART tasks."* Retrieved December 13 (2003): 2011.

User Acceptance Criteria

- User story can be translated into user acceptance criteria
- Behavior-driven development (BDD) approach
  - Vs. Test-driven development (TDD) approach
- Structure of user acceptance criteria
  - Given: the initial context of the scenario
  - When: the event that triggers the scenario
  - Then: the expected outcome
- Tools
  - Cucumber, Behave, etc.

User Acceptance Criteria Example

User Story

As a store owner,
I can add items back to inventory when they are returned,
so that I can track inventory

User Acceptance Criteria

Scenario: Refunded items should be added to inventory
Given that a customer previously bought a black sweater from me and I have three black sweaters in inventory,
when they return the black sweater for a refund,
then I should have four black sweaters in inventory

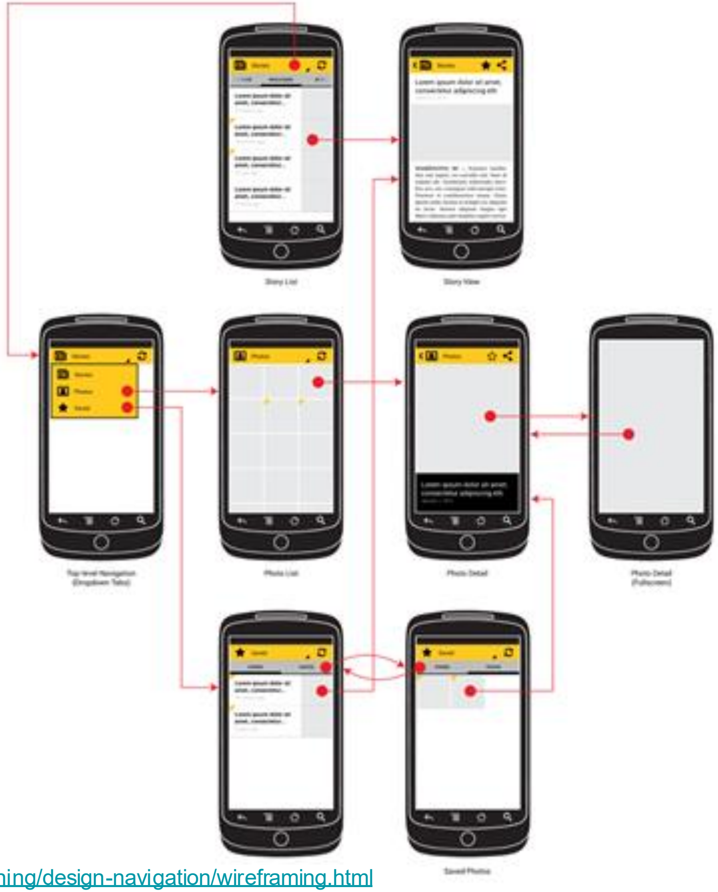
Wireframe

- Set of skeleton UIs and interactions
  - Illustrate how an application works

- Wireframe includes
  - Content hierarchy
  - Space distribution
  - Possible app user actions
  - App features
  - Transitions between app pages

Source: https://stuff.mit.edu/afs/sipb/project/android/docs/training/design-navigation/wireframing.html

Wireframe: Example



Source: https://stuff.mit.edu/afs/sipb/project/android/docs/training/design-navigation/wireframing.html

Specifications

- An explicit set of **requirements** to be satisfied by a material, product, system, or service[1]
    - Precise
    - Covering all circumstances

- Documenting requirements for various stakeholders
    - clients, domain experts, users, programmers, etc.

[1] *Form and Style of Standards, ASTM Blue Book.* ASTM International. 2012.

Good Specifications

- Adequate
  - Properly state the problem or requirements
- Consistent
  - Requirements should not conflict with each other
- Complete
  - Implementation satisfying all specified properties must exist
- Unambiguous
  - Must be interpreted in single way
- Minimal
  - Do not include requirements irrelevant to the problem
- Communicable
  - Readable and comprehensible

Source: Lamsweerde, Axel van. *"Formal specification: a roadmap."* Proceedings of the Conference on the Future of Software Engineering. 2000.

Types of Specifications

- Informal specifications
    - Written in **natural language**
    - Widely used but suffer from various problems
- Formal specifications
    - **Mathematics**-based methods
    - Hard to write and understand
- Semi-formal specifications
    - Middleground of informal and formal specifications
    - Structured analysis with **graph** notation

Informal Specifications

- Problems of informal specifications
  - Omissions
  - Ambiguities
  - Contradictions
- Disliked by academics and industries
- Most widely used

The restaurant has two tables named table A and table B. All the tables can accommodate up to and including four diners[1].

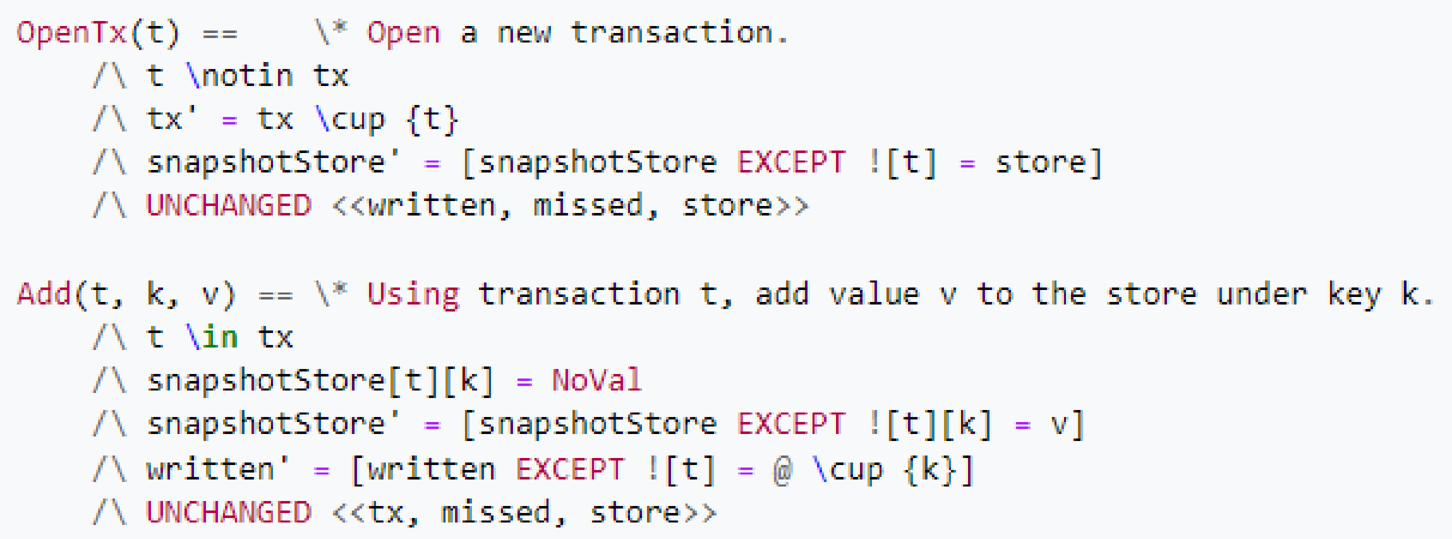
- $A+B\leq4$?
- $A\leq4$ and $B\leq4$?

[1] Kantorowitz, Eliezer. "Verbal use case specifications for informal requirements elicitation." *Proceedings of the 29th Annual European Conference on Cognitive Ergonomics*. 2011.

Formal Specifications

- Precise
- Hard to understand
- High overhead

[1]

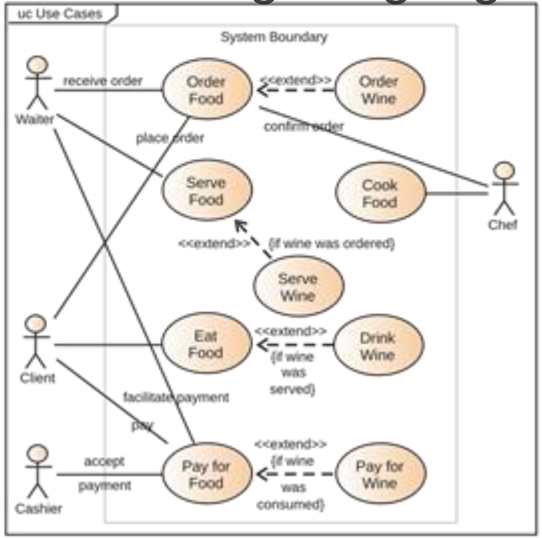


```
OpenTx(t) ==    \* Open a new transaction.
    /\ t \notin tx
    /\ tx' = tx \cup {t}
    /\ snapshotStore' = [snapshotStore EXCEPT ![t] = store]
    /\ UNCHANGED <<written, missed, store>>

Add(t, k, v) == \* Using transaction t, add value v to the store under key k.
    /\ t \in tx
    /\ snapshotStore[t][k] = NoVal
    /\ snapshotStore' = [snapshotStore EXCEPT ![t][k] = v]
    /\ written' = [written EXCEPT ![t] = @ \cup {k}]
    /\ UNCHANGED <<tx, missed, store>>
```

[1] https://en.wikipedia.org/wiki/TLA%2B

Semi-Formal Specifications

- More precise than informal specifications
- More straightforward than formal specifications
- Boxes-and-arrows diagram
  - Written with Unified Modeling Language (UML)[1]



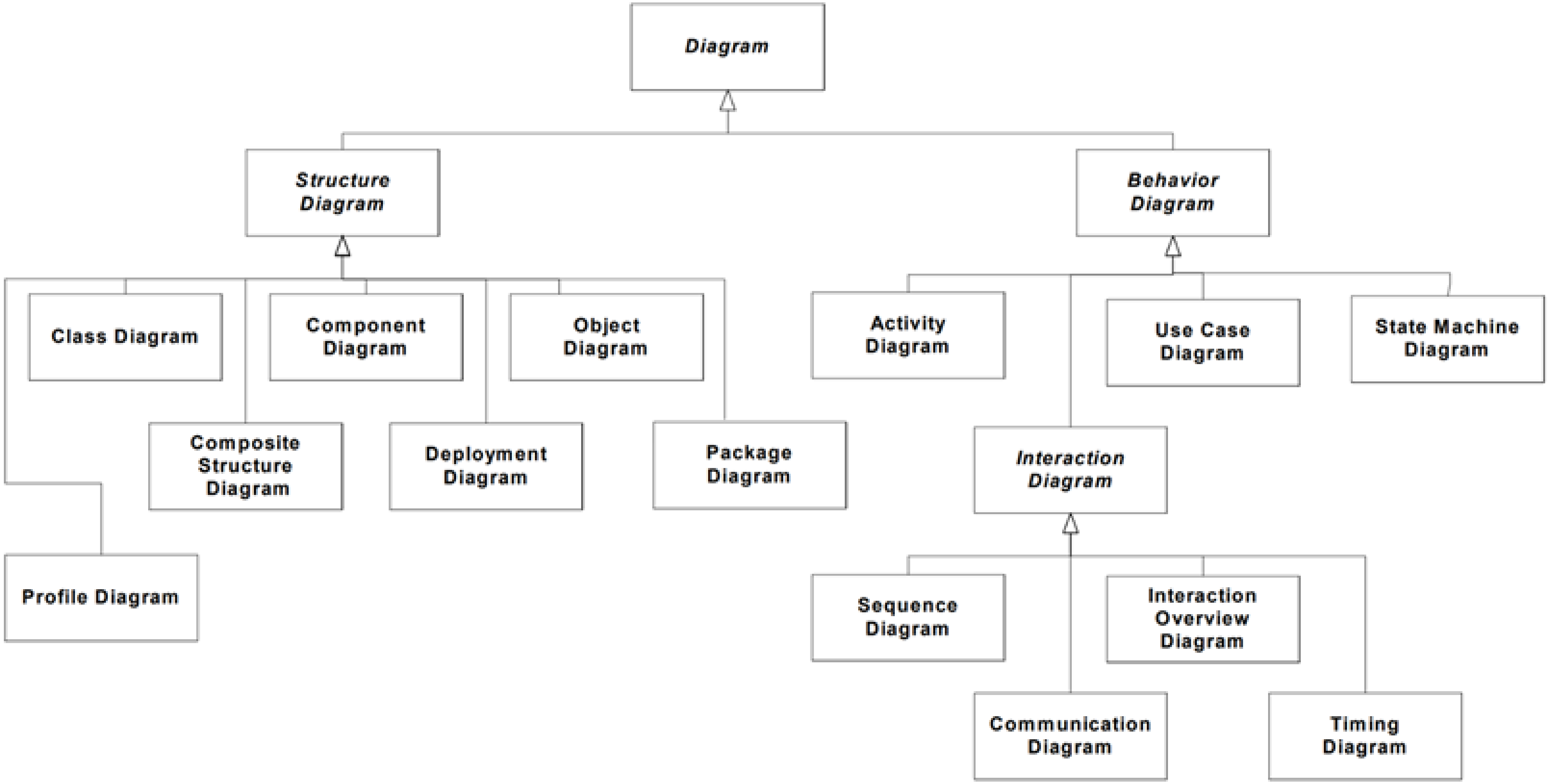
[1] https://en.wikipedia.org/wiki/Use_case_diagram

Unified Modeling Language (UML)

- General modeling language
- Standard of visualizing the design
- Two categories of diagrams
  - Structure diagrams: static aspect of the system
  - Behavior diagrams: dynamic aspect of the system

Source: Unified Modeling Language User Guide, The (2 ed.). Addison-Wesley. 2005. p. 496. ISBN 0321267974.

Diagrams of UML



Source: "OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.4.1". Object Management Group. Retrieved 9 April 2014.

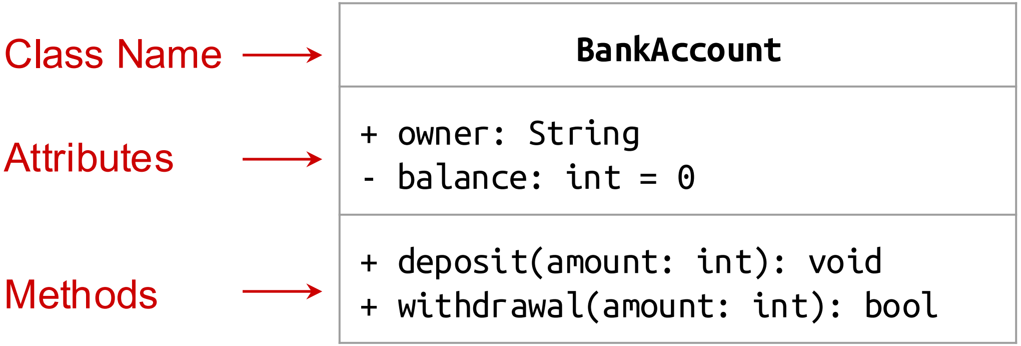
Class Diagram

- One type of static structure diagram
- Describes the structure of a system
  - Classes
  - Attributes
  - Methods
  - Relationships
- Object-oriented modeling

Source: https://en.wikipedia.org/wiki/Class_diagram

Class Diagram - Members

- Each class is depicted as a box



Class Name

Attributes

Methods

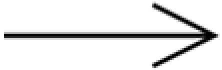
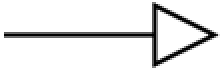
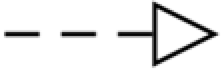
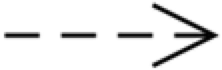
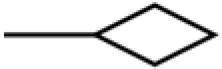
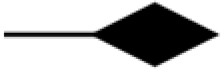
```
BankAccount

+ owner: String
- balance: int = 0

+ deposit(amount: int): void
+ withdrawal(amount: int): bool
```

- Member visibility
  - +: Public
  - -: Private
  - #: Protected
  - ~: Package

Source: https://en.wikipedia.org/wiki/Class_diagram

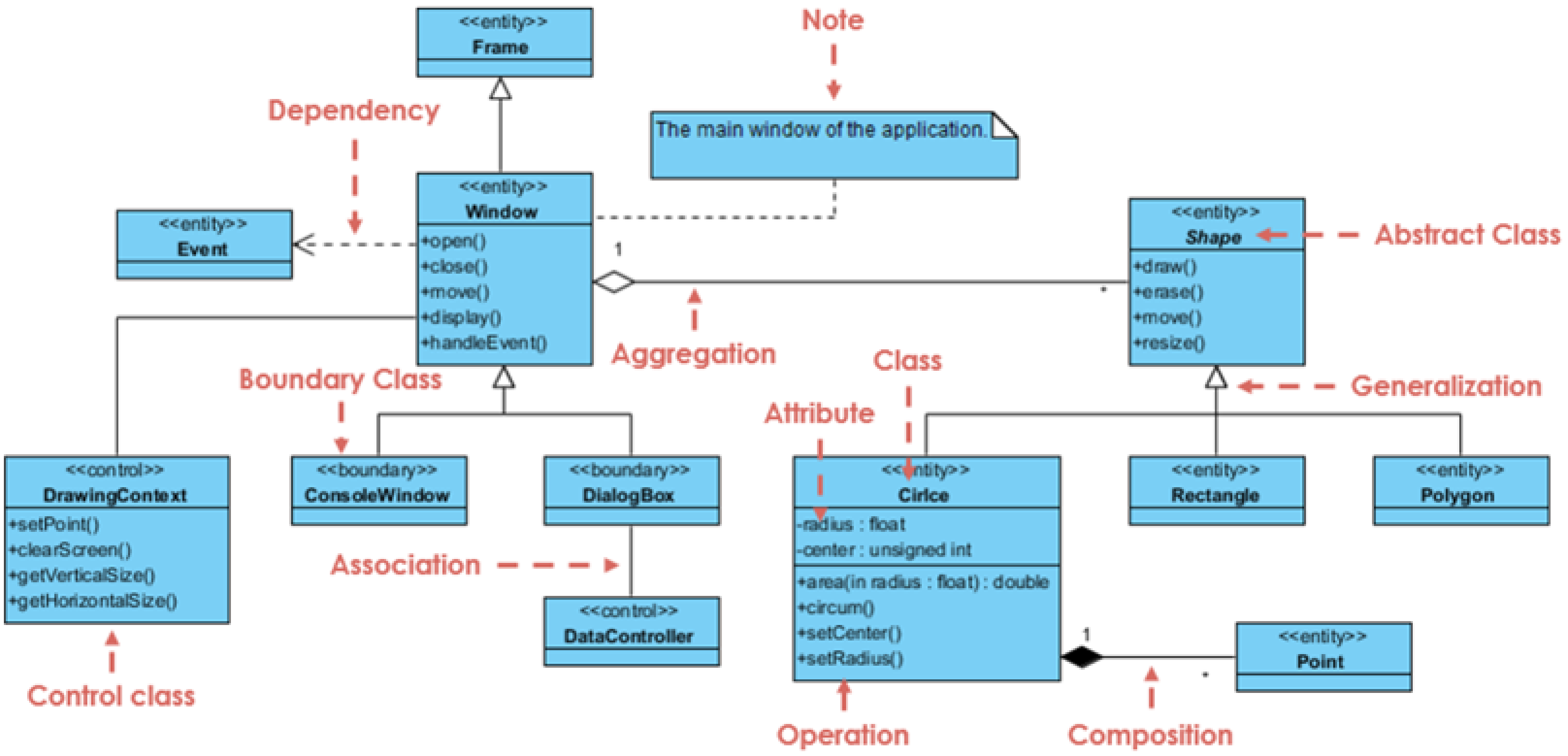
Class Diagram - Relationships

- Relationships are distinguished by the shape of the line

Line Type	Relationship Type	Description
A  B	Association	Class A is aware of class B
A  B	Inheritance	Class A extends class B
A  B	Realization	Class A implements interface B
A  B	Dependency	Class A use objects of class B, but not as member
A  B	Aggregation	Class A belongs to class B, but not only
A  B	Composition	Class A belongs only to class B

Source: https://en.wikipedia.org/wiki/Class_diagram

Class Diagram Example



Source: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

Documentations for Our Project

- Requirements
  - User stories
  - Wireframe for UI and interactions
  - Description of non-functional requirements

- Specification
  - System architecture diagram and APIs
  - Class diagrams for frontend and backend
  - Data models (ER diagram/model schema) if you use database

Summary

- Requirements are about user’s need
  - Elicitation → Analysis → Documentation
  - User stories
  - Wireframe

- Specifications are documents of requirements and system
  - Informal, formal, semi-formal methods
  - UML is widely used tool for semi-formal specification

Thank You.
Any Questions?