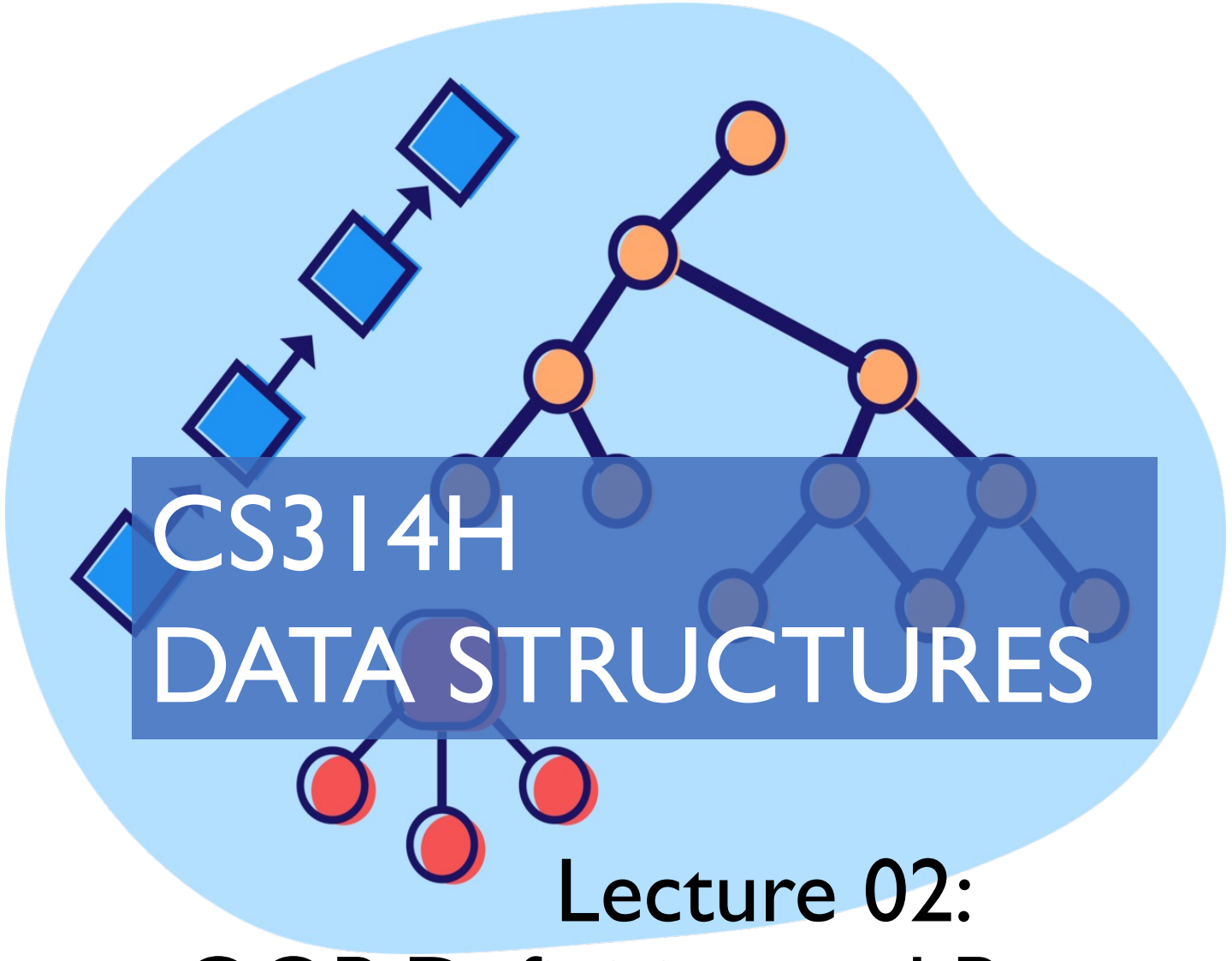

CS314H
DATA STRUCTURES

Lecture 02:
OOP Definition and Properties

Mikyung Han

Please, interrupt and ask questions AT ANY TIME !

Reminders

- Welcome Survey (Canvas)
- Syllabus Reading via Perusall (Canvas)
  - Make 4-5 meaningful comments
  - Ask/answer questions
  - Up-vote
- Assignment 0 (Ed Discussion)
- Start reading Ch3/Ch4
- Prog1 coming up
  - Read Prog 1 Spec before coming to discussion on Friday
  - Due next Thursday

Collaboration vs Cheating

- What is different?

Never share your code either offline or on-line!

Outline

1. Administrative
2. Challenges in Software Development and Management

Physical Construction vs Software Construction

What are the fundamental differences?

So really… which one is more difficult?

MS Copilot

🔍 Which Is More Difficult?

- **Software construction** is often considered **more intellectually complex** due to:
  - Abstract design and logic.
  - Rapidly changing technologies.
  - Difficulty in visualizing and validating behavior.
- **Physical construction** is often considered **more logistically and physically demanding**, with:
  - Higher stakes for safety.
  - More rigid constraints.
  - Less room for error once building begins.

Chat GPT

⚖️ **Which is more difficult?**

- **Physical construction** is more difficult in terms of *logistics, safety, and cost of mistakes*.
- **Software construction** is more difficult in terms of *complexity, flexibility, and never-ending evolution*.

A famous line in software engineering:

> *"Building software is easy; building software that works, is maintainable, and meets real needs is the hard part."*

✅ If we're talking **one-time difficulty** → building a skyscraper is probably harder.
✅ If we're talking **long-term difficulty** → maintaining and evolving software is harder, because it never stops changing.

Google Gemini

Which is More Difficult?

While both are challenging, many experts consider **software construction more difficult** due to its abstract nature and inherent complexity. Physical projects operate within a well-understood framework of physics and established engineering principles. Software projects, however, have a higher degree of interconnectedness, where a small change can have unpredictable and widespread effects. This abstract complexity, combined with the pressure for constant updates and feature additions, makes software projects highly prone to failure.

Software is non-linear

- What could small change (*i.e.* a bit flip) lead to?
- Testing

This non-linearity makes it difficult to build and maintain software

Software is constantly changing

- Why?

If software is (large, complex,)
non-linear, and constantly changing

- How should we design our software?

- How should programming languages be like?

This is the motivation for OO Design and OOP Languages

Top Challenges building/maintaining software

- Code reuse
- Facilitating changes to software
- Integrating modules
- Understanding the system as a whole

Let's see how OO Design and OOP Languages can help

Outline

1. Administrative
2. Challenges in Software Construction and Management
3. OO Model Definition

Two key ingredients in OO Model. What are they?

- Data
- What to do with the data

Two key ingredients in OO Model

They have many synonyms. Let's name some

- Data:
- Operation:

What is more important in OO Model?

Choose one

- Data
- Operation

What is more important in OO Model?

Choose one

- **Data**
- Operation

The Object-Oriented Model

- Encapsulates all program _____ inside of objects that can be accessed through _____ defined on these objects

The Object-Oriented Model

- Encapsulates all program _states_ inside of objects and can be accessed through _operations_ defined on these objects

- What does it focus on? Data >> Operation

Is this good? Why this way?

Let's compare with what is NOT OO paradigm

Imperative Programming

- Consists of set of commands for computer to perform
  - Focuses on how a program operates step-by-step
- Directly manipulates states
- What does it focus on? Data << Operation

Imperative style vs OO style

- Imperative (Direct data manipulation)

```java
public class Main {
    public static void main(String[] args) {
        int counter = 0;

        // Explicit instructions
        counter = counter + 1;
        counter = counter + 1;

        System.out.println("Counter value: " + counter);
    }
}

```

- OOP (use instance method)

```java
class Counter {
    private int value;

    public void increment() {
        value++;
    }

    public int getValue() {
        return value;
    }

public class Main {
    public static void main(String[] args) {
        Counter c = new Counter();
        c.increment();
        c.increment();

        System.out.println("Counter value: " + c.getValue()
    }
}

```

Another example: Imperative/Procedural Programming

```
1   //some code snippet written in procedural way
2   double area(Object shape) {
3       if (shape instanceof Circle) {
4           Circle circle = (Circle) shape;
5           return Math.PI * circle.radius * circle.radius
6       } else if (shape instanceof Rectangle) {
7           Rectangle rectangle = (Rectangle) shape;
8           return rectangle.width * rectangle.height;
9       } else if (shape instanceof Square) {
10          Square square = (Square) shape;
11          return square.size * square.size;
12      }
13
14      throw new IllegalArgumentException("Unknown shape");
15  }
```

Any better?

```java
//Is this any better?
double area2(Object shape) {
    if (shape instanceof Circle) {
        Circle circle = (Circle) shape;
        return Math.PI * circle.getRadius() * circle.getRadius();
    } else if (shape instanceof Rectangle) {
        Rectangle rectangle = (Rectangle) shape;
        return rectangle.getWidth() * rectangle.getHeight();
    } else if (shape instanceof Square) {
        Square square = (Square) shape;
        return square.getSize() * square.getSize();
    }
    throw new IllegalArgumentException("Unknown shape");
}
```

Write down why this is bad (At least 4 reasons!)

Why Bad?

Imperative style vs OO style

- Imperative (Direct data manipulation)

```java
public class Main {
    public static void main(String[] args) {
        int counter = 0;

        // Explicit instructions
        counter = counter + 1;
        counter = counter + 1;

        System.out.println("Counter value: " + counter);
    }
}

```

- OOP (use instance method)

```java
class Counter {
    private int value;

    public void increment() {
        value++;
    }

    public int getValue() {
        return value;
    }

public class Main {
    public static void main(String[] args) {
        Counter c = new Counter();
        c.increment();
        c.increment();

        System.out.println("Counter value: " + c.getValue()
    }
}

```

OO Model: Data >> operation

How can this help with challenging software development?

OO Model's stance (Data is more important!)

- **Protects data integrity**
- **Promotes self-contained objects**
- **Better abstraction to real world**
  - Define what an object is (inheritance)
  - What it has (its fields)
  - What it does (its operations on each field)
- **Has clear interface (API) between two objects**
  - API: public operations that can be called by clients

Now that we talked about OO Model let's talk about OO Languages

Let's see how OO Language implements OO Model

Outline

1. Administrative
2. QoD Discussion
3. Challenges in Software Construction and Management
4. OO Model
5. OO Languages

What makes a language an OO Language?

- E
- I
- P

Defining characteristics of OO Languages

- Encapsulation
- Inheritance
- (Subtype) Polymorphism

What is Encapsulation?

Encapsulation

How does Java provide encapsulation?

- Via class! Wrap data/operation inside a class
- Give each of them access modifiers

Modifier	Within the same class	Within the same package	In subclasses (same or other packages)	From other packages
`public`	Yes	Yes	Yes	Yes
`protected`	Yes	Yes	Yes	No
`default`	Yes	Yes	No	No
`private`	Yes	No	No	No

Encapsulation

- Information Hiding Principle (Dave Parnas, 1972)

  - A client of a module should have ALL info needed to use the module, and NOTHING more

  - The implementor should have ALL info needed to implement the module correctly, and NOTHING more.

Why this is good?
How does encapsulation help with 4 challenges in software?

Encapsulation – how can it help with …

- Code reuse?
- Facilitate changes?
- Integrating modules?
- Understanding system as a whole?

Discuss with your neighbors and share

Resources

- A/B Testing – an example of why software is non-linear
- Factory Pattern – Separation of concern

Backup Slides