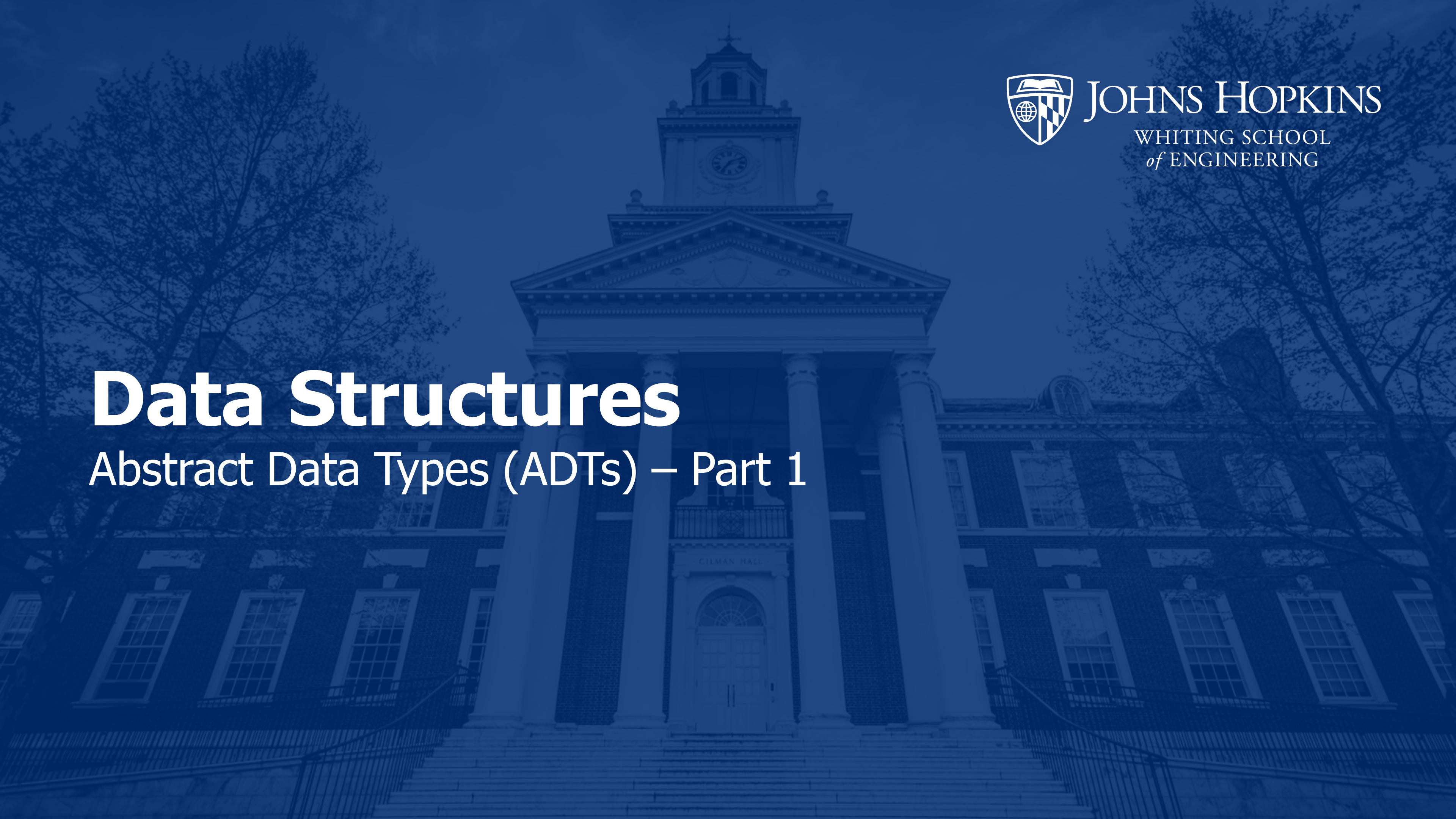


JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Data Structures

Abstract Data Types (ADTs) – Part 1

Abstract Data Types (ADTs)

A functional approach to describing information storage and access.

- No standardized approach.
- Not directly supported by language
- Requires discipline by programmer

Object Oriented Programming (OOP)

- Three Precepts:
  - Inheritance
  - Polymorphism
  - Encapsulation
- Not a goal of this course.

OOP: Polymorphism

- Java supports pseudo-polymorphism
  - e.g. operator overloading
  - e.g. templates

OOP: Encapsulation

- *Information hiding*
- *Access control*
- Supported in C++ via classes.
- Supported in Java via classes, interfaces and packages
- Public versus private declarations control access

Data Abstraction

Data abstraction requires us to divorce

1) data and how it is stored

from

2) the way it is used

Data Abstraction (cont.)

- User of data does not need to know
  - How it is stored or
  - How it is organized

- The user only concerned with
  - How can information be used
  - How can I access data

ADTs (1)

- ADT is **A**bstract **D**ata **T**ype
- An ADT is opaque
- We can't see what's inside (implementation)
- We only see what goes in and comes out. (interface)

ADTs (2)

In this course we will wear two hats.

01 **User Hat**

- we are outside the black box

02 **Programmer Hat**

- we are inside the black box

ADTs: Example (1)

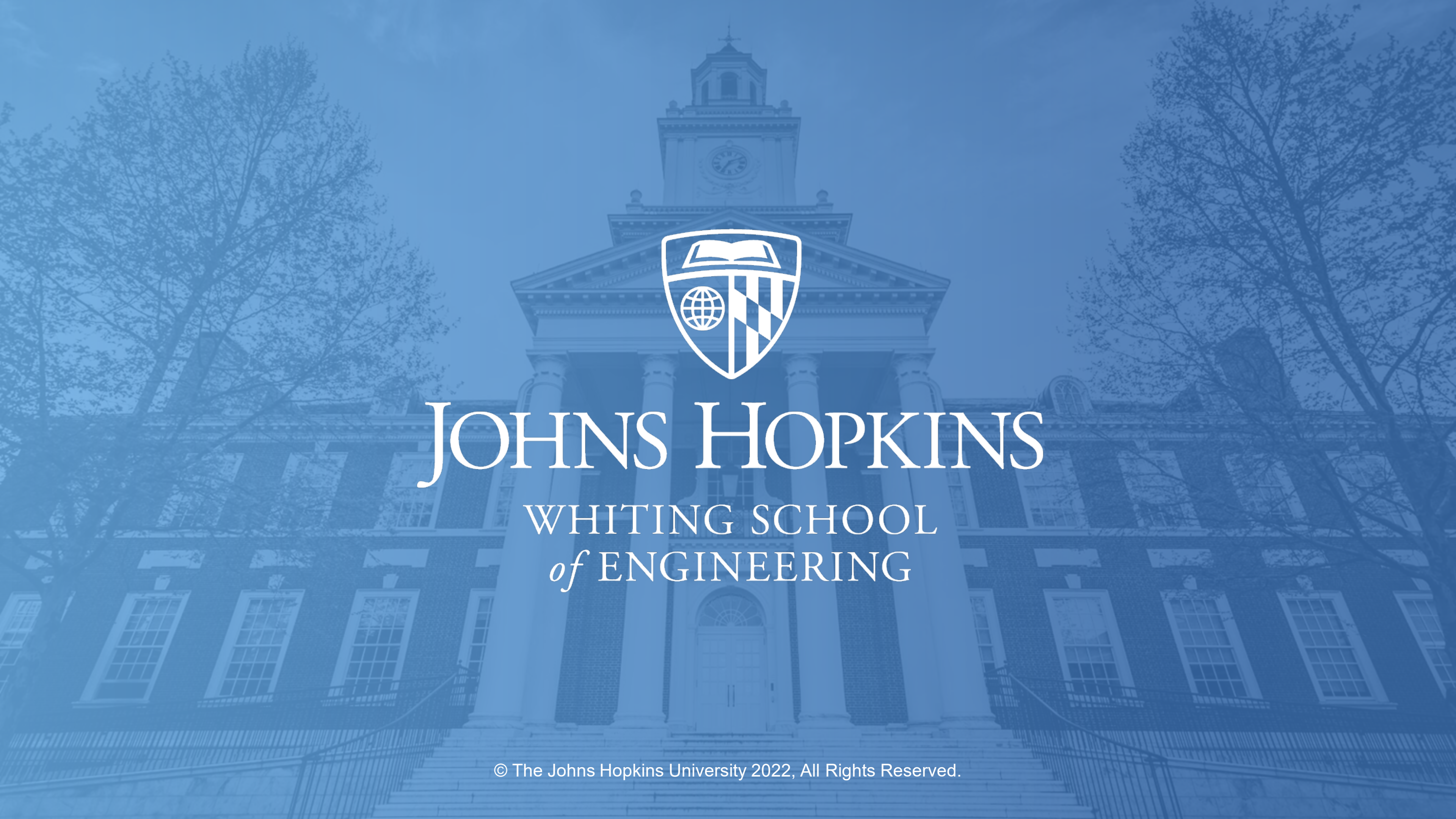
Consider a clock.

- Single function - return time of day
- Lot of implementations:
  - Battery operated,
  - 60 cycle AC
  - Pendulum,
  - Water driven

ADTs: Example (2)

Consider a clock.

- Some implementations impose restrictions.
  - Pendulum has more space requirements
  - AC power requires electrical source
- Some implementations are better choice
- Situationally dependent


JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING