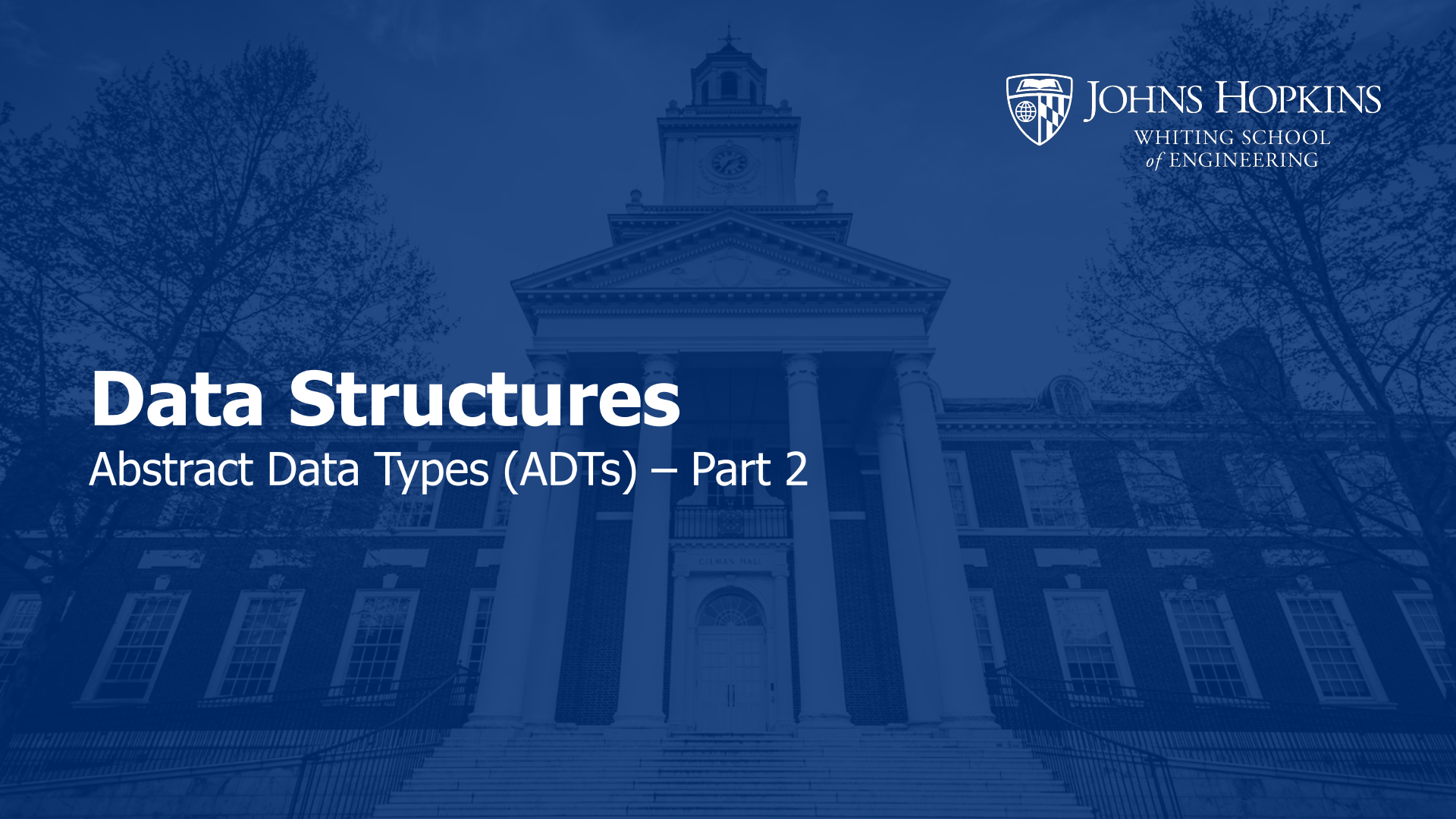



# Data Structures

## Abstract Data Types (ADTs) – Part 2

# ADTs (3)

## 01 User Hat

- Only consider the functionality
- How is information used/ accessed

## 02 Programmer Hat

- Consider implementation tradeoffs
- Consider how it is stored/organized

# ADTs: Example (3)

**Problem:** Design a new model of automobile.

**Designer**

outside the black box

**Engineer**

inside the black box

# Designer

- Thinks about how vehicle will look
  - Set a trend?
  - Be part of the crowd?
- How will vehicle appeal to a particular market segment

**Problem:** Design a new model of automobile.

# Engineer

- Thinks about implementation
- How will the parts fit together?
- Can existing factory tooling be used for all or part of manufacture?
- Costs?

**Problem:** Design a new model of automobile.

# Example: Engine (1)

- Designer identifies requirements/priorities
- Fuel efficiency/performance
- Size or weight restrictions
- Must be internal combustion engine
  - Existing societal infrastructure
  - Eliminates battery/electric power options

# Example: Engine (2)

- Engineer Identifies implementations and necessary tradeoffs
- Piston-based engine
  - Complicated
  - Gasoline powered
  - Diesel powered
- Wankel rotary engine

# Example: Engine - Gasoline

- Range of fuel economy
- Better performance
- Less pollution than diesel
- Improve fuel efficiency with alcohol mixes

# Example: Engine - Diesel

- Good fuel economy
- Strong low-end torque
- Poor acceleration
- Difficult to start in cold weather
- Fuel is less readily available

# Example: Engine - Wankel

- Simple, easy to maintain
- Poor fuel economy
- Turbo charger
  - Improves the fuel economy
  - Increases complexity

# ADTs: Specifying an ADT

- Good method is to write as a class.
- Use preconditions, postconditions, invariants
- No function bodies needed.

# ADT Format

- ADT *Name* is
  - Data: Describe the nature of the data and any initialization.
  - Methods
    - $Method_1$
      - Input: Data from the client.
      - Precondition: Necessary state of the system (what needs to be true) before executing the operation
      - Process: Actions performed with the data.
      - Postcondition: State of the system (what needs to be true) after executing the operation.
      - Output: Data values returned to the client.
    - $Method_2$ …
    - $Method_n$ …
- end ADT *Name*

# ADT Example (1)

ADT Dice

- Data
  - A count, N, of the number of dice in a single toss, the sum of the toss, and a list of the N tossed die values. Values of a toss range from 1 to 6. Sum ranges from 1N to 6N.
- Methods
  - (next slide)

# Methods

*Toss*

- **Input:** None
- **Precondition:** None
- **Process:** Toss the dice and compute the sum.
- **Postcondition:** The sum contains the sum of the dice on the toss, and the list identifies the value of each die in the toss.
- **Output:** None

*DieTotal*

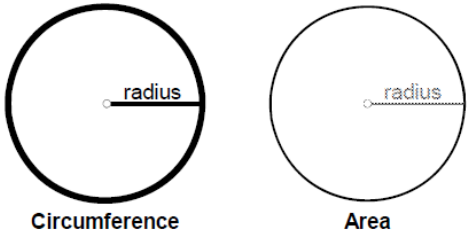
- **Input:** None
- **Precondition:** None
- **Process:** Retrieve the value of the variable which specifies the sum for the most recent toss.
- **Postcondition:** None
- **Output:** Return the total of the dice for the most recent toss.

*Display Toss*

- **Input:** None
- **Precondition:** None
- **Process:** Print the list of dice values for the most recent toss.
- **Output:** None
- **Postcondition:**

End ADT Dice

# ADT Circle is




- **Data**:
  - A non-negative real number specifying the radius of the circle, initialized to a non zero real radius.
- **Methods**:

*Area*

- **Input:** None
- **Precondition:** None
- **Process:** Computer the area of the circle.
- **Postcondition:** None
- **Output:** Return the area.

*Circumference*

- **Input:** None
- **Precondition:** None
- **Process:** Computer the circumference of the circle.
- **Postcondition:** None
- **Output:** Return the circumference.

End ADT Circle

# ADT Example (2)

```
#include <iostream.h>
const float PI = 3.14152;
//declare Circle class with data and method declarations
class Circle
{
private:
float radius; //initialize to a positive value
public:
Circle (float r); //constructor
float Circumference(void) const;
float Area(void) const;
};
```

Note structural similarity between ADT and Class declaration.