Polymorphism & Real-World Applications

Welcome to Lesson 3 - Polymorphism & Real-World Application.

In this two-hour session, we'll explore polymorphism—one of the most powerful concepts in object-oriented programming—and see how it applies to real-world scenarios.

Throughout this lesson, we'll examine how polymorphism allows different classes to use the same method names while implementing them differently, making our code more flexible and intuitive. We'll combine theory with hands-on activities to reinforce these concepts.

CN **da Carlo Nicolò**

Warm-Up Activity

Video Game Examples

How is polymorphism used in video games where different character classes share the same method names?

Character Actions

Consider how various characters might implement an attack() method differently while maintaining the same interface.

Think-Pair-Share

Discuss with a partner how this concept makes game development more efficient and code more maintainable.

Key OOP Concepts: Polymorphism

Definition

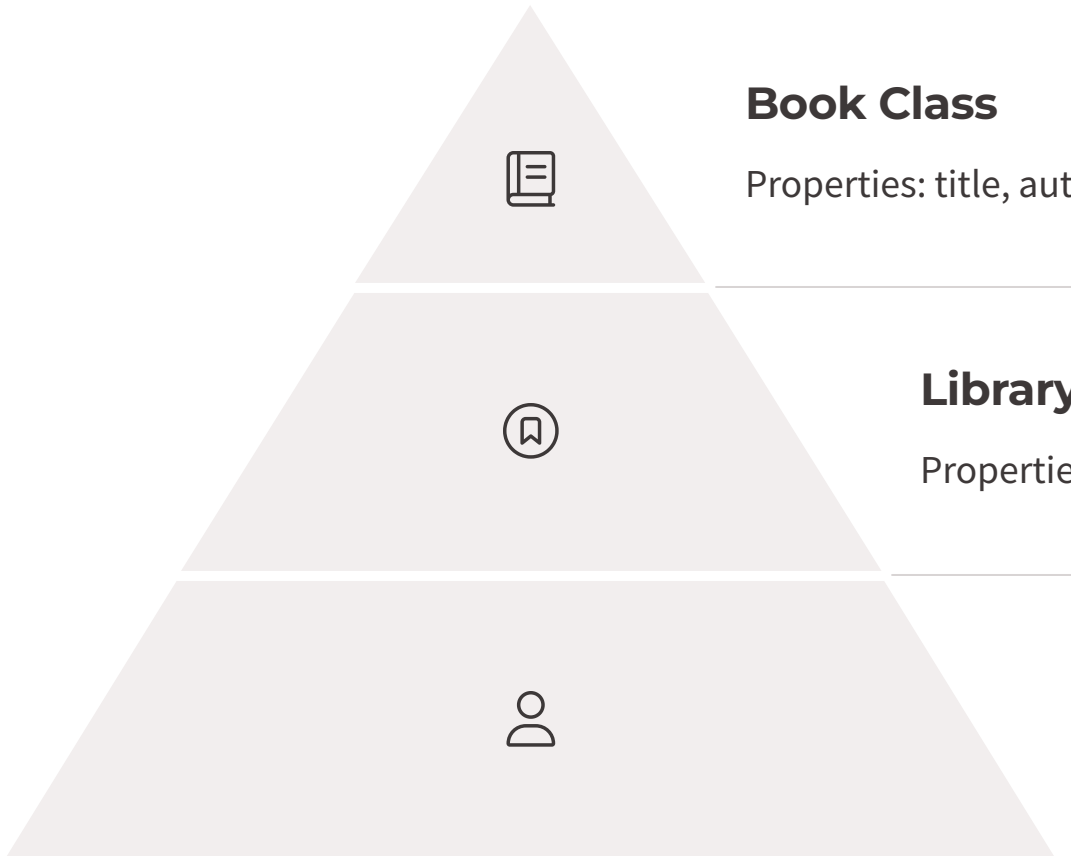
Polymorphism is the ability for different classes to use the same method name but implement it differently. This creates a consistent interface while allowing for specialized behavior.

For example, both an Archer and Knight class can have an attack() method, but each implements it according to their character type.

```
class Archer:
    def attack(self):
        print("Shoots arrow!")

class Knight:
    def attack(self):
        print("Slashes sword!")
```

Group Project: Library System



Book Class

Properties: title, author, Methods: get_info()

Library Class

Properties: books, Methods: add_book(), borrow_book()

User Class

Properties: name, borrowed_books, Methods: borrow(), return()

In this 50-minute activity, you'll work in groups to design these classes. Focus on how polymorphism might be used when different types of books implement get_info() differently, or when different user types (students, faculty) might have different borrowing privileges.

Presentation & Feedback

Present Your Design

Each group will have 3-5 minutes to present their Library System design, focusing on how they implemented polymorphic methods.

Demonstrate Code

Show a brief code example of how your polymorphic methods would work in practice.

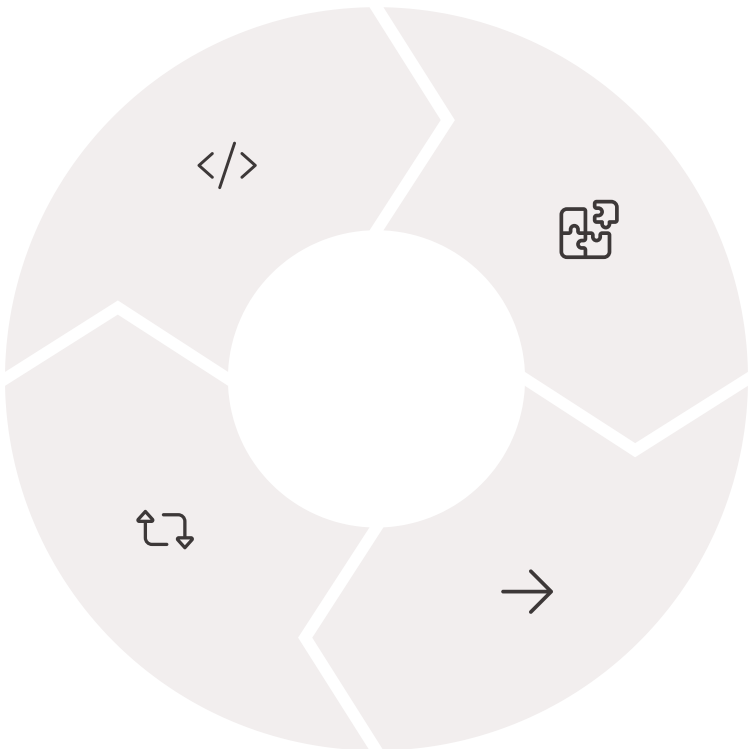
Receive Feedback

Classmates and instructor will provide constructive feedback on your design choices and implementation.

Polymorphism in Action

Define Interface

Create common method signatures across classes



Implement Differently

Each class provides its own implementation

Extend Easily

Add new classes without changing existing code

Use Polymorphically

Call methods through base class or interface

Wrap-Up Discussion

Code Flexibility

How does polymorphism make your code more adaptable to change? Consider how new classes can be added without modifying existing code.

Maintenance Benefits

Discuss how polymorphism reduces code duplication and creates cleaner interfaces between components.

Real-World Applications

Identify examples of polymorphism in software you use daily, from apps to operating systems.

Design Challenge: School System

Students

Properties: name, ID, courses

Methods: enroll(), viewGrades()

Teachers

Properties: name, ID, subjects

Methods: assignGrade(), createAssignment()

Courses

Properties: name, code, students

Methods: addStudent(), getAverage()

Your challenge is to draw a UML diagram for this school system, focusing on how polymorphism might be implemented. Consider how a Person class might be used as a base class for both Students and Teachers.

Homework Assignment

Research Task

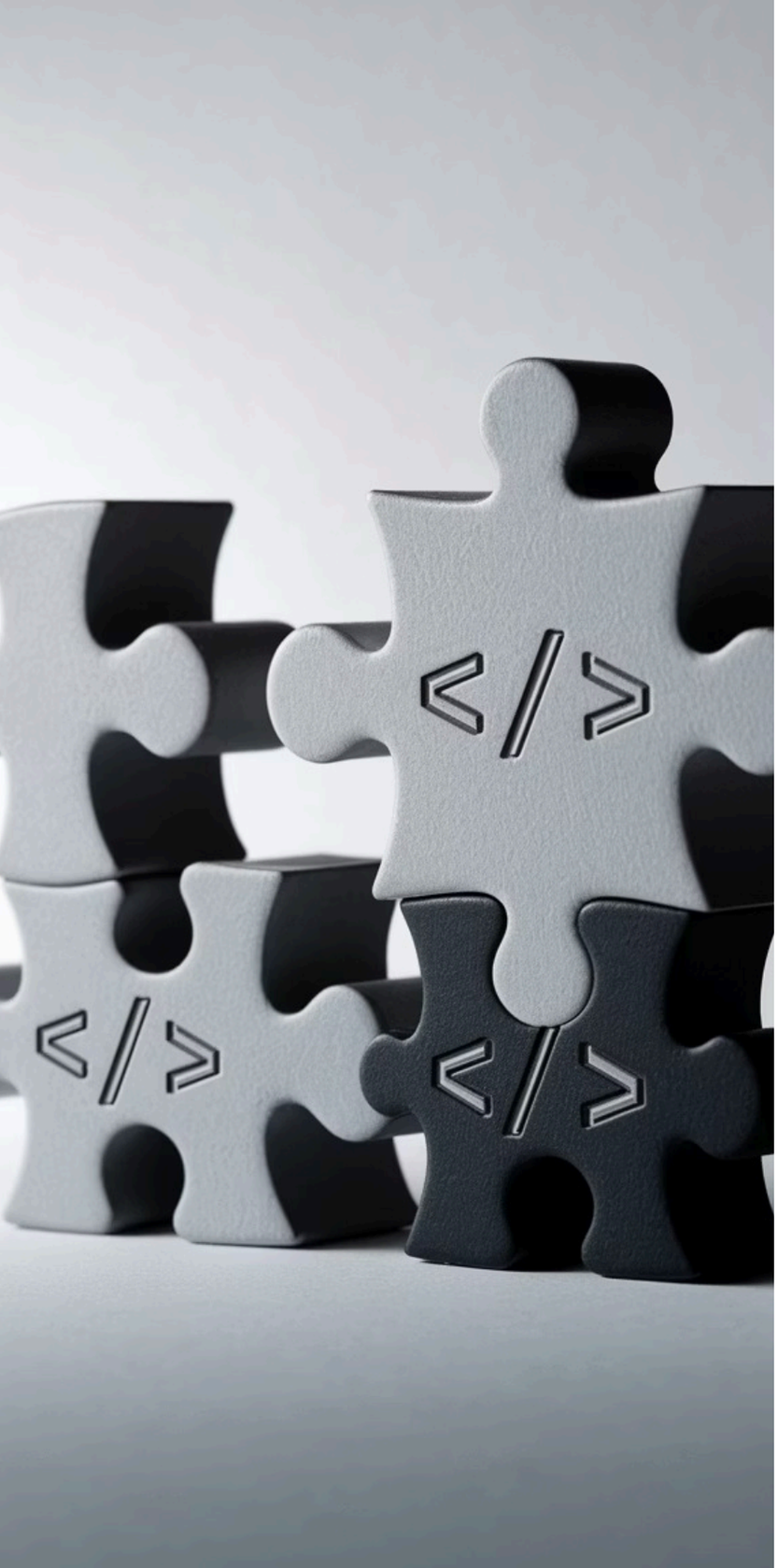
Compare object-oriented programming in Python versus Java, with special attention to how polymorphism is implemented in each language.

Coding Exercise

Extend the Library system we designed in class with a search() method that works polymorphically across different book types.

Documentation

Create a brief write-up explaining your implementation choices and how polymorphism enhances your solution.



Key Takeaways

1

Same Interface

Polymorphism allows different classes to respond to the same method calls, creating consistent interfaces.

2

Different Implementations

Each class can provide its own specialized behavior while maintaining the same method signatures.

3

Code Flexibility

Systems become more extensible as new classes can be added without changing existing code.

Remember that polymorphism is one of the four pillars of object-oriented programming, alongside encapsulation, inheritance, and abstraction. Mastering these concepts will make you a more effective programmer in any OOP language.