Clevis: Command LinE Vector graphics Software
Group 20 Project Presentation

Group member:

Tseung Cheuk Kit (24132668d)

Cheng Chun Ngai (24094912d)

So Lok Hei Daniel(24102213d)

Lam Tsz Chung Oscar(24081241d)

System Overview

Supported Shapes

- Rectangle
- Circle
- Line
- Square
- Group (composite)

Key Features

- Shape creation & manipulation
- Grouping/ungrouping
- Intersection detection
- Bounding box calculation
- Z-index management

Three-Layer Architecture

Three-Layer Architecture

Presentation Layer

- Application.java
- CLI handler
- GUI viewer
- Logger

Business Logic

- ShapeManager
- CommandParser
- QueryHandler
- Operations

Model Layer

- Shape interface
- Rectangle
- Circle, Line
- Square, Group

Model Layer

- Shape interface
- Rectangle
- Circle, Line
- Square, Group

Design Choice: Inheritance & Polymorphi

Shape Interface

```
interface Shape {
String name();
BoundingBox bbox();
void move(double dx, double dy);
boolean intersects(Shape other);
Shape pick(int newZ);
}
```

Polymorphic Behavior

✓ Each shape implements intersection differently

✓ Uniform interface for all operations

✓ Runtime type determination

Benefits

✓ Easy to add new shapes

✓ Consistent API

✓ Testable components

Example:

Polymorphic Dispatch Pattern

Circle.intersects()

→ Circle vs Circle: distance formula

→ Circle vs Rectangle: closest point

→ Circle vs Line: point-to-line distance

→ Circle vs Group: delegate

Rectangle.intersects()

→ Rect vs Rect: overlap test

→ Rect vs Circle: closest point

→ Rect vs Line: segment intersection

→ Handles containment cases

Group Intersection (Composite Pattern)

Tests all pairs: O(n×m) complexity, early termination on first hit

Object-Oriented Design Benefits

Reusability

✓ Shape Interface: Reused across all shape types
✓ Helper Methods: ShapeQueryHandler utilities shared by all shapes
✓ Group Pattern: Composite pattern enables recursive operations

Maintainability

Scalability

✓ New Shapes: Just implement Shape interface
✓ New Operations: Add without modifying existing code
✓ Extensibility: Open/Closed Principle applied

Encapsulation ensures each class has single responsibility. Changes isolated to specific components.

The End