# Allocation

## Resources(准备题目答案)

## Entity(用户实体+数据访问：题目和答案)

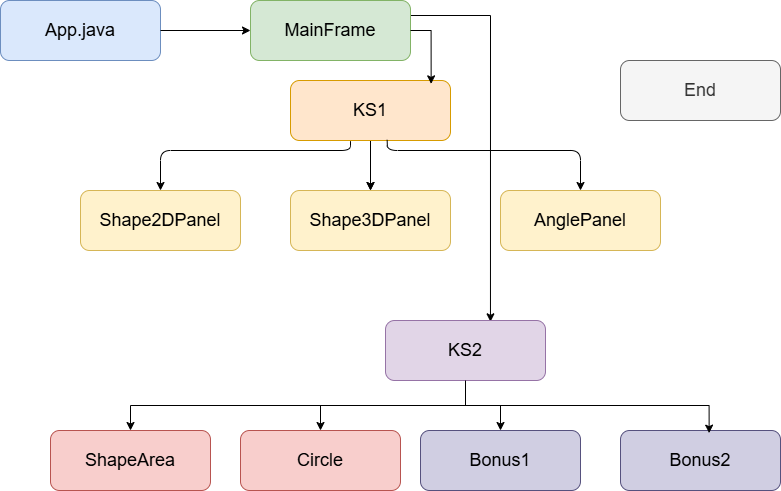
## Service（业务逻辑层）

## UI实现（用户交互层）

## UI画图设计（没要求做，但是为了凑分工，可以对着成品画几个原型图，假装我们有UI设计这个工种存在）

# Design of the system

## Flowchart



1. Project Initiation

The project starts with [App.java], which serves as the entry point of the application, responsible for initializing and launching the program.

2. Main Interface Display

In [App.java], the [MainFrame] is called. This class represents the main framework of the application and is used to host various functional panels in the subsequent stages.

3. Phase 1 Functionality (KS1)

Within [MainFrame], the user enters the first functional module, known as `KS1`.

Functional Branches in KS1:

Shape2DPanel: Handles operations related to two-dimensional shapes.

Shape3DPanel: Manages tasks involving three-dimensional objects.

AnglePanel: Deals with functionalities related to angles.

4. Phase 2 Functionality (KS2)

From any of the three panels in `KS1` ([Shape2DPanel], [Shape3DPanel], or [AnglePanel]), the user can proceed to the second stage of functionality, referred to as `KS2`.

Functional Branches in KS2:

ShapeArea: Calculates the area of different shapes.

Circle: Provides specific operations related to circles.

Bonus1 and Bonus2: Additional bonus modules offering extra features.

5. End of the Process

Once all operations are completed, the process reaches the `End` state, indicating that the program has finished execution.

6. Module Functions:

The KS1 phase primarily provides foundational functionalities for handling shapes and angle-related tasks.

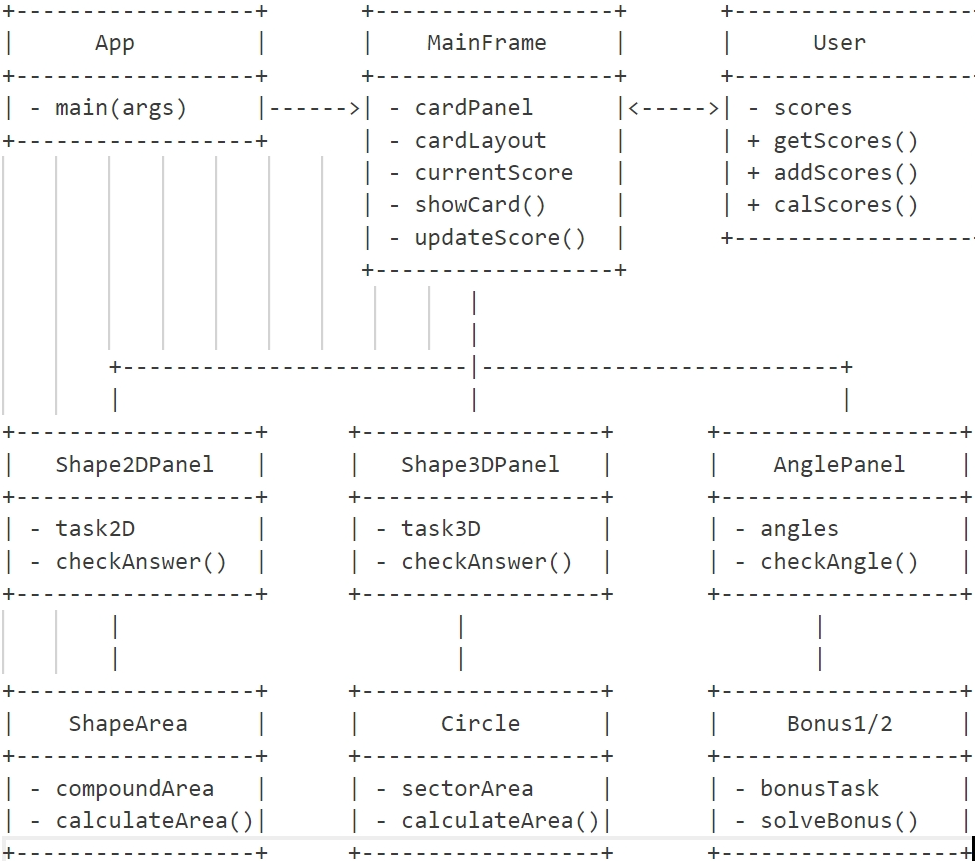
The KS2 phase builds upon KS1 by offering deeper and more specific computations of shape properties, along with additional features.

This flowchart clearly illustrates the modular design of the project and the step-by-step implementation of its functionalities, aiding in the understanding of each stage’s purpose and the overall structure of the project.

7. Summary of the Workflow:

The primary workflow begins with [App.java], transitions through [MainFrame] into the `KS1` phase, where users can choose to work with [Shape2DPanel], [Shape3DPanel], or [AnglePanel]. Following this, users may proceed to the `KS2` phase for more detailed calculations such as shape areas, circle-related operations, or bonus functions. The flow ultimately concludes at `End`.

## Main Classes



### Project Structure and Functionality Description

1. App Class

Function: [main(args)]: The entry point of the program, responsible for starting the application.

1. MainFrame Class

Attributes:

[cardPanel]: A card panel used for switching between different task panels.

[cardLayout]: A card layout manager.

[currentScore]: The current score of the user.

Methods:

[showCard()]: Displays a specific task panel.

[updateScore()]: Updates the user's score.

1. User Class

Attributes:

[scores]: A list of the user's scores.

Methods:

[getScores()]: Retrieves the user's scores.

[addScores()]: Adds new scores to the user's score list.

[calScores()]: Calculates the total score of the user.

1. Shape2DPanel Class

Attributes:

`task2D`: A 2D shape task.

Methods:

[checkAnswer()]: Checks if the user's answer to the 2D shape task is correct.

1. Shape3DPanel Class

Attributes:

`task3D`: A 3D shape task.

Methods:

[checkAnswer()]: Checks if the user's answer to the 3D shape task is correct.

1. AnglePanel Class

Attributes:

`angles`: An angle task.

Methods:

`checkAngle()`: Checks if the user's answer to the angle task is correct.

1. ShapeArea Class

Attributes:

`compoundArea`: A compound area.

Methods:

[calculateArea()]: Calculates the area of a shape.

1. Circle Class

Attributes:

`sectorArea`: The area of a sector.

Methods:

[calculateArea()]: Calculates the area of a circle or sector.

1. Bonus1/2 Class

Attributes:

`bonusTask`: A bonus task.

Methods:

`solveBonus()`: Solves the bonus task.

### Project Workflow Description

1. Application Startup:

The application starts through the [main(args)] method in the [App] class.

1. Main Interface Display:

The [MainFrame] class initializes and displays the main interface, which includes a [cardPanel] for switching between different task panels.

1. Task Panel Display:

Based on the user's selection, the [showCard()]method in [MainFrame] displays the corresponding task panel ([Shape2DPanel], [Shape3DPanel], or [AnglePanel]).

1. Task Execution and Verification:

The user completes tasks within the task panels.

The respective task panels ([Shape2DPanel], [Shape3DPanel], or [AnglePanel]) call the appropriate verification methods ([checkAnswer()]or `checkAngle()`) to check the correctness of the user's answers.

1. Score Update:

If the answer is correct, the [updateScore()]method in [MainFrame] updates the user's score.

1. Score Management:

The [User] class manages the user's scores, providing methods such as [getScores()], [addScores()], and [calScores()]to retrieve, add, and calculate scores.

1. Additional Features:

The [ShapeArea] and [Circle] classes provide functionality for calculating the areas of shapes.

The `Bonus1/2` classes offer additional task-solving features.

Summary:this project is a graphical learning application that tests and enhances users' knowledge of shapes through various task panels (2D shapes, 3D shapes, and angle tasks). The core logic is managed by the [MainFrame] class, while user scores are maintained by the [User] class. Specific task verification and shape calculations are handled by the respective panel classes and service classes. The design is modular with clear responsibilities, making it easy to extend and maintain.

# Design Highlights Analysis

## Adaptive Design for Colorblind Users

### Diverse Color Schemes

The Shapeville application implements three distinct color schemes to accommodate users with different visual abilities:

Standard Vision Mode: Uses conventional color combinations including vibrant blues, cyans, and greens to create an engaging interface for typical users.

Red-Green Colorblind Mode: Specifically designed for deuteranopia/protanopia users, avoiding red-green contrasts and primarily using blue and yellow-orange combinations.

Blue-Yellow Colorblind Mode: Designed for tritanopia users, employing purple and red as primary hues to prevent blue-yellow confusion.

### Adaptive Background Images

The application provides specially designed background images for each color scheme, automatically adjusting based on the user's selected visual mode.

### Semantic Color Definitions

The design utilizes semantic color naming (e.g., "primary", "success") rather than hard-coded color values, enabling seamless theme switching while maintaining consistent information hierarchy across all color schemes.

### Dynamic Color Switching Mechanism

The application features one-click color scheme switching with comprehensive UI synchronization:

Automatic button text updates indicating current scheme

Dynamic background image adjustment

Instantaneous UI element recoloring

## Child-Friendly Interaction Design

### Specialized Button Components

Custom-designed buttons incorporate multiple child-friendly elements:

Rounded Corners: 20px border radius for soft edges

Playful Typography: Comic Sans MS font for informal appeal

Oversized Targets: Larger touch areas for developing motor skills

Concise Labels: Simple, direct text for easy comprehension

### Interactive Animation Effects

Engaging micro-interactions enhance child engagement:

Smooth scaling animations on hover

Immediate visual feedback mechanisms

Attention-sustaining motion design

### Visual Progress Tracking

Clear status visualization helps children understand task completion:

Color-coded task indicators (completed/incomplete)

Achievement reinforcement through visual cues

Progress visualization supporting learning motivation

# User-manual

## Introduction

Shapeville is an interactive educational application designed to help children aged 5-10 learn geometry concepts through engaging activities. The software offers a colorful, accessible environment where young learners can explore 2D shapes, 3D shapes, angles, areas, and more.

## System Requirements

Operating System: Windows 7 or newer, macOS 10.12 or newer, Linux

Java Runtime Environment (JRE) 8 or newer

Minimum Screen Resolution: 1024 x 768

Memory: 256MB RAM minimum

## Getting Started

1. Use IDEs such as vscode, idea, eclipse to run the program "App.java".

2. The main menu will appear showing two learning paths:

Key Stage 1 (KS1): Basic shape and angle recognition (for younger children)

Key Stage 2 (KS2): More advanced concepts like area and perimeter (for older children)

## Application Interface

The Shapeville interface consists of:

Top Area: Shows the "Color Mode" button for accessibility settings

Main Content Area: Displays the current learning activity

Bottom Status Bar: Shows your progress score and task completion status

Navigation Buttons: Appear at the bottom of most screens to help you move between sections

## Accessibility Features

Shapeville is designed to be accessible to all children, including those with color vision deficiencies:

1. Color Schemes: The application provides three different color schemes that can be toggled using the "Color Mode" button:

Normal Vision: Standard color palette with vibrant colors

Red-Green Colorblind: Special color scheme for those with red-green color vision deficiency

Blue-Yellow Colorblind: Adapted color scheme for blue-yellow color vision deficiency

2. Each time you press the "Color Mode" button, the application will cycle to the next color scheme.

## Learning Modules

**Key Stage 1 (KS1)**

**2D Shapes**

View and learn various 2D shapes

Identify shapes shown on screen by typing their names

Receive instant feedback on your answers

You have 3 attempts for each shape

**3D Shapes**

Learn about three-dimensional shapes and their properties

Identify 3D shapes from their images

Similar to 2D shapes, you have 3 attempts for each answer

**Angles**

Learn different types of angles (acute, right, obtuse, straight, reflex)

Practice identifying angle types

Interactive angle visualizations help reinforce concepts

**Key Stage 2 (KS2)**

**Shape Area**

Learn how to calculate areas of various shapes (rectangles, triangles, parallelograms, trapeziums)

Practice solving area problems with randomly generated parameters

Visual representations help understand area formulas

**Circle**

Learn about circle properties, including area and circumference

Practice calculations using both radius and diameter

Visual illustrations demonstrate the relationship between dimensions and calculations

**Bonus Modules**

**Compound Area**: Calculate areas of complex shapes made from multiple simple shapes

**Sector Area**: Learn about circle sectors and how to calculate their areas

## Scoring System

Shapeville tracks your progress with a points-based scoring system:

Each correct answer earns points based on:

Task difficulty (Basic or Advanced)

Number of attempts used (fewer attempts = more points)

Basic level activities award 1-3 points per correct answer

Advanced level activities award 2-6 points per correct answer

Your current score appears in the progress bar at the bottom of the screen

Task completion status is shown with colored indicators

## Tips for Teachers/Parents

1. Start with KS1: For young children (5-7), begin with the Key Stage 1 modules

2. Progression: Move to Key Stage 2 once the child is comfortable with shape recognition

3. Supervision: Younger children may need assistance with typing answers

4. Explanations: Use the visual representations as teaching aids to explain concepts

5.Accessibility: Adjust the color scheme if your child has any color vision deficiency

## Ending a Session

1. Click the "End" button in the bottom right corner of the screen

2. A summary dialog will show the total points earned during the session

3. Click "OK" or press the "Enter" to close the application

## Troubleshooting

Application won't start: Ensure you have Java installed on your computer

Text appears cut off: Try resizing the window to see if it helps

Colors don't match descriptions: Try cycling through the color modes using the "Color Mode" button

Thank you for using Shapeville to explore the world of geometry with your young learners!

# Utilisation of AI

## Colorblind-Friendly Color System Implementation

During the development of the Shapeville geometry learning application, AI-assisted tools played a crucial role in designing and implementing a comprehensive colorblind adaptation system. Through interaction with AI tools, I was able to:

Research colorblindness types and characteristics: AI helped me understand the visual characteristics of red-green colorblindness (most common) and blue-yellow colorblindness, providing scientific basis for the adaptation design.

Generate color combinations: Based on AI-provided colorblind-friendly color theory, I implemented three complete color schemes:

Blue as a substitute for green in red-green colorblind mode

Yellow-orange for warning indicators

Purple as primary color and cyan for success indicators in blue-yellow colorblind mode

Verify color contrast: AI tools assisted in verifying the contrast and distinguishability of each color scheme, ensuring clear differentiation of interface elements for different colorblind users.

Implement dynamic switching mechanism: With AI suggestions, I created a semantic color reference system enabling seamless switching between different color schemes.

AI tools significantly enhanced my understanding of colorblind users' needs and helped design a professional color adaptation system that would otherwise require extensive specialized research and testing.

## Java Drawing Functionality (Geometric Visualization)

For the geometric visualization module, AI tools notably improved development efficiency and code quality:

Complex drawing algorithms: AI assisted in implementing sophisticated Java 2D drawing code for various shapes including parallelograms, triangles, rectangles, and trapezoids.

Mathematical formula visualization: AI tools helped design visual representations of area calculation formulas, including:

Fractional notation

Arrow markers

Dynamic calculation results

Making teaching effects more intuitive.

Coordinate transformation and scaling: AI provided algorithms for handling different scale values and coordinate transformations, ensuring correct display of shapes under various parameters.

Color coordination and consistency: AI assisted in creating drawing schemes that maintain recognizability across different color modes, ensuring graphic elements remain clear in colorblind modes.

## Python Angle Generation Script

For the angle learning module, AI tools assisted in developing a Python script for batch angle image generation:

Matplotlib drawing code generation: AI helped write Python code using Matplotlib library to create precise angle visualizations.

Special angle handling: AI provided specialized visualization code for different angle types (acute, right, obtuse, straight, and reflex angles), each with unique representations.

Batch generation and file management: AI implemented looping structures and filesystem operations for batch image generation, enabling easy creation of 36 angle images from 0°to 350°.

Visual consistency: AI ensured all generated angle images maintained consistent visual style and educational elements, unifying the application's overall design language.

AI-assisted tools played a pivotal role in the Shapeville project, particularly in three core aspects:

The use of these tools not only shortened the development cycle but also notably improved the application's accessibility and educational effectiveness, enabling Shapeville to better meet the geometry learning needs of children aged 5-10. AI input proved essential for achieving the project's educational goals and inclusive design principles.

# Reflection on your experience

## My First Java GUI Project Experience

At first, this project really surprised me because it wasn’t a typical CRUD (Create, Read, Update, Delete) project like I expected. I felt a little lost at the beginning, but after carefully reading the Product Requirements Document, I started building the project structure. Instead of using the MVC (Model-View-Controller) architecture common in web projects, I went with a simpler three-layer design:

Data Layer – Handles user data, question images, and answers.

Service Layer – Processes the core functions.

UI Layer – Manages the GUI (Graphical User Interface).

## What Went Well

AI is advancing so fast! For this project, I used Cursor (an AI coding assistant) and Claude 3.7 Sonnet (a powerful AI model). They helped me with a lot of things—from project structure to solving tricky problems. Sometimes, AI gave me amazing solutions, but other times, it was a bit unpredictable. Still, I learned a lot about AI-assisted programming.

I also gained real-world experience with GUI programming. Before, I only knew about events and listeners from PowerPoint slides. Now, I actually used .addEventListener() with anonymous inner classes and lambda functions. It turns out, Java GUI has a lot in common with front-end web development (HTML, CSS, JS)—just different syntax, but the core ideas are similar!

## What Could Be Improved

High Coupling Between Layers.At first, I didn’t design interfaces properly, so different parts of the project depended too much on each other. Next time, I’ll use interfaces to reduce dependencies.

Manual Testing (No Unit Tests). I tested everything by hand, running the program over and over. That’s not efficient! In future projects, I’ll learn unit testing (like JUnit) to make sure my code works correctly before running the whole program.