EBU4201 Introductory Java Programming 2024/25 Mini Project Report of Group88

Jiayang Lyu	Yucheng Tang	Zelang Wen	Hanyu Xiao	Weijia Xiao
231226130	231226141	231226222	231226244	231226233

1.Task Allocation & Team Contribution

This project was completed collaboratively by five team members, with the following specific responsibilities:

Jiayang Lyu

- Responsible for the overall system architecture design, including module division, main class design, and functional allocation.
- Oversaw the development process to ensure smooth integration between modules and system scalability.

Yucheng Tang

- Responsible for the overall UI (User Interface) design and implementation.
- Unified the visual style, optimized user interaction, and ensured the interface is attractive, user-friendly, and accessible.

Zelang Wen

- Responsible for all drawing functionalities in Task 2, Task 3, and Task 4, including angle diagrams, diagrams of four basic shapes, and circle illustrations.
- Contributed significantly to detail optimization, such as interface fine-tuning, feature improvements, and enhancing user interaction experience.

Hanyu Xiao

- Responsible for version control and cross-platform compatibility testing.
- Ensured the project could be compiled and run smoothly on different operating systems (e.g., Windows, macOS), and addressed related compatibility issues.

Weijia Xiao

- Responsible for global functionality testing and user experience optimization.
- Conducted systematic testing, identified and fixed bugs, proposed and implemented user experience improvements, and ensured the application's stability and reliability.

2. System Design

2.1 Overall System Architecture

The Shaperville project adopts a layered and modular architecture, consisting of three main layers:

- User Interface Layer (View): All user interface classes are located in the src/view package, including the main menu, task panels (such as 2D/3D shape identification, angle recognition, area calculation, compound shapes, and sector challenges), progress bars, and feedback areas.
  The UI is implemented entirely with Java Swing, featuring a unified cartoon style and accessibility optimizations.
- Business Logic Layer (Controller/Service): This layer is mainly reflected in the control logic within each panel, such as question generation, answer validation, scoring, and progress management. For example, each task panel is responsible for generating questions, validating answers, providing feedback, and updating progress.
  The ScoreManager singleton manages the global score, and ScoringUtil calculates the score for each question.
- Data Model Layer (Model): All data structures for questions, scores, and types are located in the src/model package. Key classes include the abstract Question class (with ShapeQuestion and AngleQuestion as concrete implementations), Score, ScoreManager, ScoringUtil, and the enums ShapeType and AngleType.
  The ImageLoader utility class is used for image loading and scaling.

2.2 System Flow Chart

The following flow chart illustrates the main user flow of the Shaperville application:

1. The user launches the application and enters the main menu (MainMenuPanel).
2. The user selects a task (e.g., 2D/3D shape identification, angle recognition, area calculation, compound/sector challenge).
3. The system switches to the corresponding task panel, automatically generates and displays questions.
4. The user answers the questions, receives instant feedback, and sees progress and score updates.
5. Upon completing a task, the system displays the score for that task, and the user can return to the main menu or continue with other tasks.
6. The user can click “End session” at any time in the main menu to view the total score and reset progress.

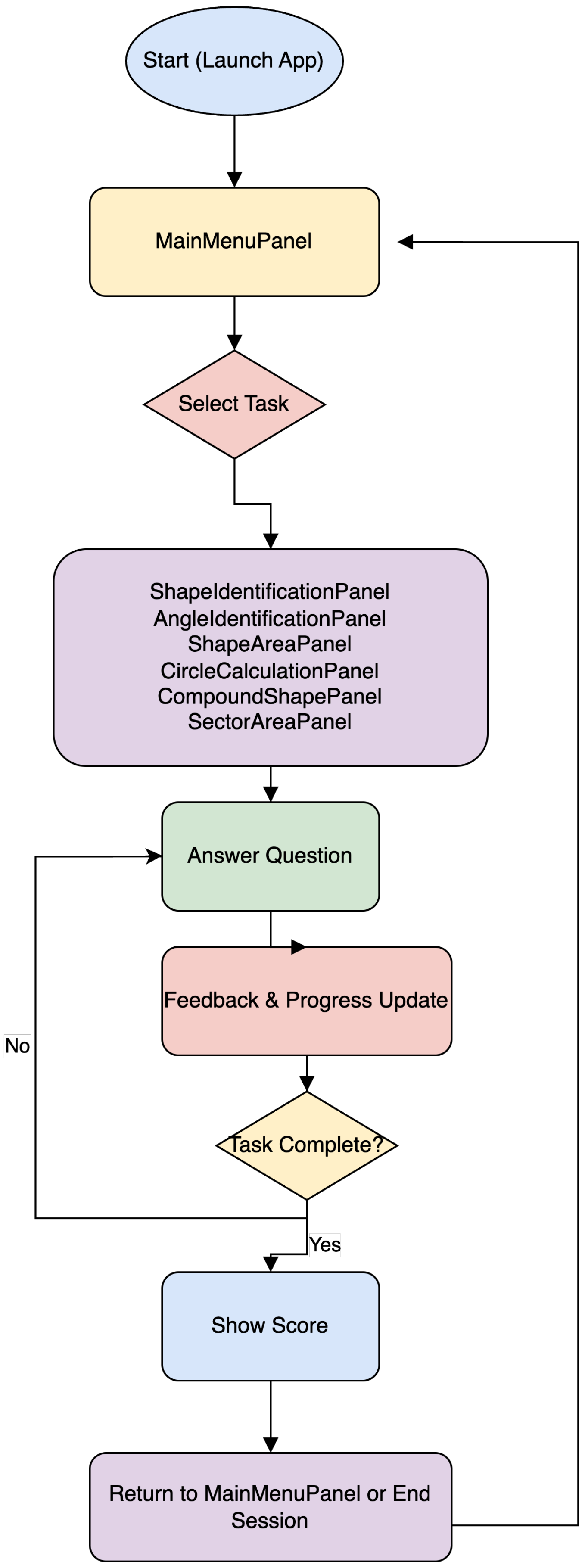


Figure 1: System Flow Chart

2.3 Class Diagram

The following class diagram shows the main classes and their relationships in the Shaperville project:

- Entry Point: ShapervilleApp: The main entry point, responsible for initializing the window and loading the main menu.
- Main Menu and Task Panels:
  - MainMenuPanel: The main menu, responsible for task selection, global progress display, and session management.
  - Task panels such as ShapeIdentificationPanel, AngleIdentificationPanel, ShapeAreaPanel, CircleCalculationPanel, CompoundShapeSelectionPanel, CompoundShapePanel, SectorSelectionPanel, and SectorAreaPanel all extend JPanel and handle their respective UI and logic.
- Question and Scoring Model:
  - Question (abstract): Defines common properties for all question types.
  - ShapeQuestion, AngleQuestion: Concrete implementations for specific question types.
  - Score: Records the score and number of attempts for each question.
  - ScoreManager: Singleton for managing the global score.
  - ScoringUtil: Utility for calculating scores.
- Utilities and Enums:
  - ImageLoader: Utility for image loading and scaling.
  - StyleUtil: Utility for consistent UI styling.
  - ShapeType, AngleType: Enums for all supported shape and angle types.

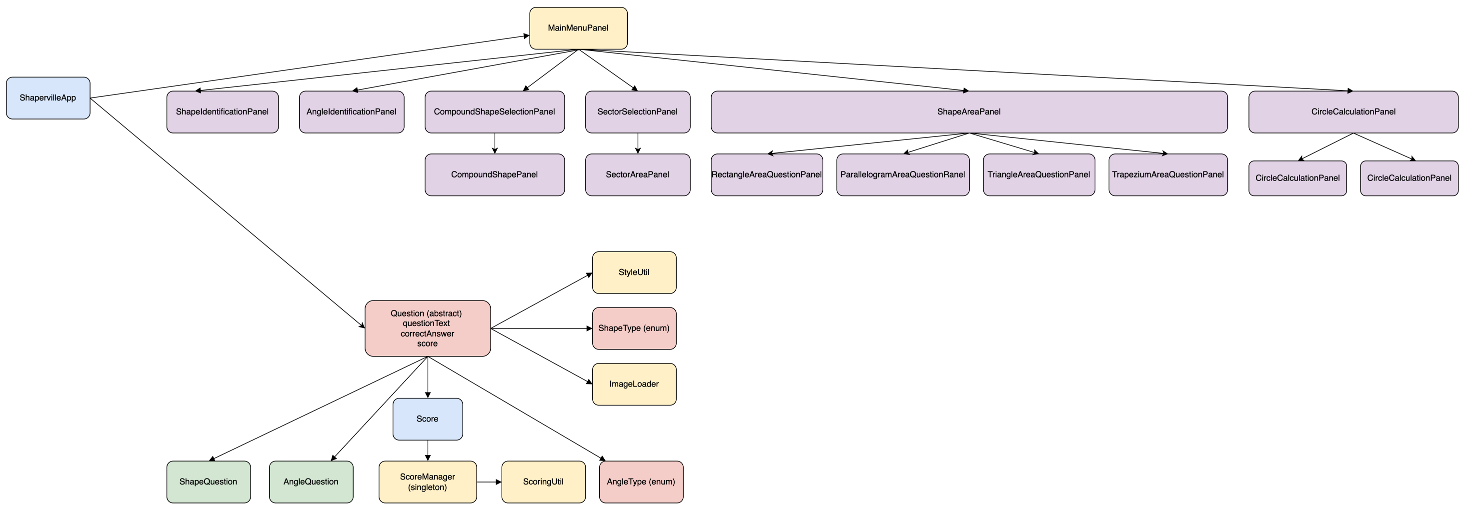


Figure 2: Class Diagram

2.4 Data Flow and Module Interaction

- The main menu is responsible for navigation and global progress display. Clicking a task button switches to the corresponding panel.
- Each task panel handles question generation, answer validation, scoring, and progress management for its task.
- When answering, the panel calls ScoreManager to update the global score and uses ScoringUtil to calculate the score for the current question.
- All images are loaded via ImageLoader, and UI styles are unified through StyleUtil.
- Task completion status is shared between the main menu and panels via static methods or variables.

2.5 Accessibility and Extensibility

- All UI components use high-contrast color schemes, large fonts, rounded buttons, and are designed to be color-blind friendly and accessible.
- Progress bars and feedback areas use highlights and emojis, and buttons change cursor on hover for better user experience.
- The code structure is clear and modular, making it easy to add new question types or panels with minimal changes to existing code.

3. Design Justification

The design of the Shaperville application was guided by the goals of clarity, accessibility, engagement, and ease of use, especially for younger users and those with diverse needs. Below are the key justifications for our design choices:

3.1 User Interface and Layout

- Cartoon Style & Visual Engagement: The application adopts a cartoon-like visual style, using the "Comic Sans MS" font, bright color blocks, and rounded buttons. This makes the interface friendly and appealing to children, encouraging engagement and reducing anxiety when facing mathematical tasks.
- Consistent and Unified Design: All panels and buttons follow a unified color scheme and style, managed by the StyleUtil utility class. This ensures a coherent user experience and makes navigation intuitive.
- Clear Task Segmentation: The main menu divides tasks into KS1, KS2, and Bonus sections, each with clear labels and color-coded buttons. This helps users quickly identify and access the desired module.

3.2 Task Flow and User Guidance

- Step-by-Step Progression: Each task panel guides the user through a clear, step-by-step process, with progress bars indicating completion status. For example, in Task 1 and Task 2, progress bars visually track the number of questions completed or angle types practiced.
- Input Validation and Error Messages: All user inputs are validated with clear error messages (e.g., for angle input: "from 0 to 360 degrees (excluding 0 and 360), and a multiple of 10"). This prevents confusion and guides users to correct their mistakes.

3.3 Accessibility and Inclusivity

- Large Fonts and Clickable Areas: All text uses large, legible fonts, and buttons are generously sized with ample spacing. This benefits users with visual impairments and those using touch devices
- Mouse-Friendly Interactive Elements: All interactive elements (such as buttons and text fields) provide clear visual feedback, such as changing the cursor to a hand on hover, ensuring that users can easily operate the application with a mouse.

- Color-Blind and Attention-Deficit Friendly Design: The color palette is carefully chosen to ensure sufficient contrast, making it easy for users with color vision deficiencies to distinguish between different interface elements. To further support users with color blindness, we provide textual cues in all scenarios where color is used as an indicator. For example, in Task 1, the progress bar not only uses color to indicate correctness, but also displays a checkmark or cross for correct or incorrect answers. In tasks where users can select questions, completed buttons not only change color but also display the word “Completed” or a checkmark. For users with attention deficit, we adopted a minimalist design approach, which maximizes the prominence of core content on the interface and reduces distractions, helping these users stay focused on the main tasks.

4. User Manual

This section provides a step-by-step guide for using the Shaperville application. The manual is designed for non-technical users and covers all main features.

4.1 Starting the Application

- Launching the Program: After obtaining the program files, open a command line (terminal) in the project folder (i.e., EBU4201-Group88-Project) and execute the following commands to compile and start the application:

  To compile:

  ```
  javac src/ShapervilleApp.java
  ```

  To run:

  ```
  java src/ShapervilleApp
  ```

  The program will automatically enter the main menu after starting.

- Main Menu: After launching, you will see the main menu, which is divided into three sections: KS1, KS2, and Bonus. Each section contains different geometry learning tasks.

4.2 Task Overview and Navigation

- KS1 Section:
  - 2D Shape Identification: Practice recognizing common 2D shapes.
  - 3D Shape Identification: Practice recognizing common 3D shapes.
  - Angle Identification: Practice identifying different types of angles.
- KS2 Section:
  - Area Calculation of Shapes: Practice calculating the area of rectangles, triangles, parallelograms, and trapeziums.
  - Circle Calculation: Practice calculating the area and circumference of circles.
- Bonus Section:
  - Compound Shape Challenge: Solve area problems involving compound shapes.
  - Sector Area Challenge: Solve area problems involving sectors of circles.
- Progress Bar: At the bottom of the main menu, a progress bar visually displays your completion status for each task.
- End Session: Click the “End session” button at the bottom to view your total score and reset your progress.

4.3 How to Use Each Task

4.3.1 Shape Identification (2D/3D)

- Click the corresponding button in the KS1 section to enter.
- The program will randomly present four questions. For each question, a shape image will be displayed. Enter the name of the shape in the input box.
- Click “Submit” to check your answer. The system will provide instant feedback and show your progress.
- After completing all questions, you can view your total score for this task.
- Click the Home button to return to the main menu.

4.3.2 Angle Identification

- Click the “Angle Identification” button in the KS1 section to enter.
- Enter an angle value (between 0 and 360 degrees, excluding 0 and 360, and must be a multiple of 10), then click “Draw Angle” to generate the angle diagram.
- Enter the type of angle (e.g., acute, obtuse, right, straight, reflex) in the input box and click “Submit.”
- Feedback and progress will be displayed below the input box.
- Click the Home button to return to the main menu.

4.3.3 Area Calculation of Shapes

- Click the “Area Calculation of Shapes” button in the KS2 section to enter.
- Select the shape you want to calculate (rectangle, parallelogram, triangle, or trapezium) by clicking the button with its image.
- Read the given parameters, enter your answer, and click “Submit.”
- After answering, the correct formula, a diagram, and a detailed calculation process will be displayed.
- After completing all shapes, the task ends. Click the Home button to return to the main menu.

4.3.4 Circle Calculation

- Click the “Circle Calculation” button in the KS2 section to enter.
- Choose to calculate either the area or circumference of a circle.
- Read the given radius or diameter, enter your answer, and click “Submit.”
- After answering, feedback, the formula, and calculation steps will be displayed.
- Click the Home button to return to the main menu.

4.3.5 Compound Shape Challenge

- Click the “Compound Shape Challenge” button in the Bonus section to enter.
- Select a compound shape to solve.

- Read the problem, enter your answer, and submit.
- Your progress will be recorded, and completed shapes will be marked.
- After completing all shapes, the task ends. Click the Home button to return to the main menu.

4.3.6 Sector Area Challenge

- Click the “Sector Area Challenge” button in the Bonus section to enter.
- Select a sector to solve.
- Read the problem, enter your answer, and submit.
- Your progress will be recorded, and completed sectors will be marked.
- After completing all sectors, the task ends. Click the Home button to return to the main menu.

4.4 Tips and Troubleshooting

- Input Validation: If you enter an invalid answer (such as a non-numeric value or an out-of-range angle), the system will display a clear error message. Please follow the on-screen instructions to correct your input.

- Progress and Scores: Your progress and scores are automatically saved during the session. You can return to the main menu at any time.

- Progress Reset: For Task 3/4 and Bonus 1/2, if you return to the main menu in the middle of answering, the program will retain your progress (only correctly answered questions are counted as completed; unanswered or incorrectly answered questions are not counted). The next time you enter the task, you will continue from your previous progress.
  If you exit the program using the “End session” button, all progress will be reset, making it easy to start a new practice session.

5. Use of AI Tools & Prompts

This section describes how AI tools were utilized during the development of the Shaperville project, including typical application scenarios, key prompts submitted, and how the AI responses were adopted or modified.

5.1 Overview of AI Tool Utilization

During the development of Shaperville, team members used different AI tools according to their personal work habits. The main AI tools used were Deepseek and Claude. These tools mainly assisted us in the following tasks:

- Code Generation and Optimization: Generating basic code for Java Swing panels, event handling, utility classes, and optimizing code structure to improve readability and maintainability.
- Design Advice: Consulting on interface layout, accessibility best practices, and color schemes suitable for children and users with special needs.
- Debugging and Troubleshooting: Seeking advice on error messages, Swing interface compatibility, and cross-platform operation issues.
- Documentation and Report Writing: Assisting in drafting comments and project documentation.

5.2 Specific Tools and Integration Methods

- Deepseek
  We used the Deepseek-V3/R1 model directly through the official Deepseek web portal.
- Claude
  We used the Claude model via the Copilot plugin in VS Code, which allowed seamless integration of AI tools into our programming workflow.

5.3 Typical Application Scenarios and Prompt Examples

During development, we mainly used AI tools to solve various programming errors and knowledge gaps in the design process. For example, when designing the interface, we were unsure how to make it more friendly for users with color vision deficiencies and attention deficit. The AI, with its vast knowledge base, provided reasonable suggestions, some of which we adopted based on our technical capabilities. In programming, AI assisted us in implementing specific features and provided optimization suggestions at the architectural level, enabling us to build the project more scientifically. In addition, when ensuring cross-platform compatibility, AI also helped us solve many practical problems.

We have included three specific prompts and AI responses in the Appendix, where you can see the detailed scenarios and how we adopted the suggestions.

5.4 Experience with AI Tools

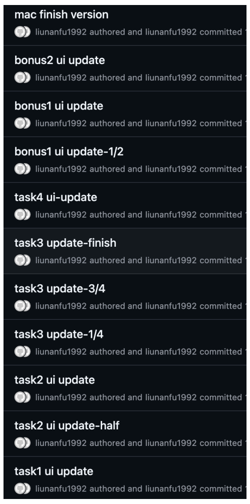
Although AI tools are now very powerful, we found that due to limitations in contextual understanding, it is still quite difficult for them to lead a project independently. When solving different problems within the same project, AI tools often exhibit self-contradictory “hallucinations.” Therefore, during our use, we actively (and necessarily) reviewed the suggestions and code provided by AI tools and made modifications to fit the actual needs of the project.

6. Reflection on Group Work

There is no AI involved here; our group will present our most direct thoughts in this section :)

This Mini-Project consists of multiple tasks. Although the questions in each task are quite simple for university students, we truly felt challenged when we had to build interactive interfaces for these questions through code—there were many macro-level and detailed programming issues that we needed to solve. Therefore, we believe this project is valuable and a good exercise. During the development process, thanks to clear division of labor, our group made very smooth progress in implementing the underlying logic and the initial design of the interactive interface. However, we soon encountered two significant problems:

1. Due to some mistakes in git version management during development, our team members faced issues where their work could not be merged together. We spent a lot of time resolving this and subsequently established new internal guidelines for using git within the group.
2. During the process of beautifying the UI, since we had already completed and fixed the initial design of the interactive interface, problems arose when we started modifying backgrounds, buttons, fonts, etc. For example, after adding a beige background, the angle diagrams drawn in Task 2 were covered by the background. Issues like this forced us to almost reconsider the implementation and design of the functionality and interface from scratch. Fortunately, after a group meeting, we agreed on a solution that required minimal changes. After verifying its feasibility in the test environment, we promptly applied the modifications to the official version.



```
mac finish version
liunanfu1992 authored and liunanfu1992 committed
bonus2 ui update
liunanfu1992 authored and liunanfu1992 committed
bonus1 ui update
liunanfu1992 authored and liunanfu1992 committed
bonus1 ui update-1/2
liunanfu1992 authored and liunanfu1992 committed
task4 ui-update
liunanfu1992 authored and liunanfu1992 committed
task3 update-finish
liunanfu1992 authored and liunanfu1992 committed
task3 update-3/4
liunanfu1992 authored and liunanfu1992 committed
task3 update-1/4
liunanfu1992 authored and liunanfu1992 committed
task2 ui update
liunanfu1992 authored and liunanfu1992 committed
task2 ui update-half
liunanfu1992 authored and liunanfu1992 committed
task1 ui update
liunanfu1992 authored and liunanfu1992 committed
```

Figure 3: This is the Git commit record for solving the second issue, and it looks very impressive, doesn't it?

Therefore, if we were to do this project again, we would definitely finalize the UI style at the same time as designing the interface sketches, so that we could choose implementation methods with fewer conflicts in advance. We would also agree on git usage conventions early on to make teamwork smoother.
In addition, we hope that next time we can receive a more accurate project specification at the very beginning of the project…

That concludes our group reflection. We would like to thank the EBU4201 module instructors for always answering our questions on the forum throughout the Mini-Project.

Sincerely from all members of Group 88,
Jiayang Lyu, Yucheng Tang, Zelang Wen, Hanyu Xiao, Weijia Xiao
May 21, 2025

Appendix: AI Usage Examples

- Example 1:
  - Model: Deepseek-R1
  - Prompt: I'm facing some design issues. How can I calculate the global score across different question modules? Please give me some suggestions and inspiration.
  - AI's Main Response: The AI suggested implementing a singleton pattern for a ScoreManager class, which would include methods for increasing, retrieving, and resetting scores. It emphasized the importance of maintaining a centralized score management system to ensure consistency across different modules.
  - Adoption: We largely adopted the AI's design proposal. However, since the AI lacked context about the project details, the provided code was not usable. We implemented the code ourselves based on the AI's suggestions.
- Example 2:
  - Model: Deepseek-R1
  - Prompt: Our project aims to be friendly to color-blind users. Can you provide some design suggestions?
  - AI's Main Response: The AI recommended using high-contrast color schemes, large and friendly fonts (such as Comic Sans MS), and avoiding red-green combinations. It also suggested incorporating alternative indicators, such as shapes or patterns, to convey information without relying solely on color.
  - Adoption: We adopted the AI's suggestions for interface style design, which is directly reflected in our program's interface.
- Example 3:
  - Model: Claude 3.7 Sonnet (via Copilot)
  - Prompt: "We are facing cross-platform issues. After migrating the project files from Mac to Windows, we cannot compile the files properly. Please help me troubleshoot the problem based on the console error messages."
  - AI's Main Response: The AI noted that the error might be due to differences in how macOS and Windows handle text files. It suggested using javac -encoding UTF-8 src/ShapervilleApp.java on a Windows machine to compile the project. The AI also recommended checking for any platform-specific dependencies or configurations that might be causing the issue.
  - Adoption: After trying the AI's solution, we successfully compiled our project files on Windows.