1. **Tile.cs** - Core data structures:

* TileColor enum (9 colors including Black for empty)
* Tile class representing individual tiles
* TilePattern class for preset patterns

1. **TileManager.cs** - Tile management classes:

* TileBoard - 8x8 grid management
* TileChecker - Row/column completion detection
* TilePurger - Clearing completed tiles with scoring

1. **TilePatternManager.cs** - Pattern management:

* All 20 tile patterns as specified in requirements
* Random pattern generation
* Pattern retrieval by ID

1. **ScoreManager.cs** - Scoring system:

* Score tracking and combo system
* Combo timeout (10 seconds)
* Score calculation with multipliers

1. **SoundManager.cs** - Audio management:

* Sound and music handling
* Placeholder for future audio implementation

1. **GameManager.cs** - Main game logic:

* Game state management (pause, game over)
* Save/load functionality
* Pattern placement logic
* Game over detection

1. **BlockBlastGame.cs** - Main game class:

* Rendering and input handling
* Game loop management
* UI drawing

1. **Program.cs** - Clean entry point:

* Only contains the main method
* Instantiates and runs the game

System Architecture Now Matches Requirements:

✅ Tileset: 2D array [x][y] containing tile values (0 = blank)

✅ TileBoard: Houses the tileset, manages the 8x8 grid

✅ TilePatternManager: Manages all 20 tile patterns with exact tileSet[x][y] syntax

✅ TileChecker: Checks tileset 2D array, returns counts, uses ArrayList

✅ TilePurger: Uses ArrayList, sets values back to zero, handles scoring/combo

✅ Game Flow: Patterns placed using top-left edge positioning