Design Choices

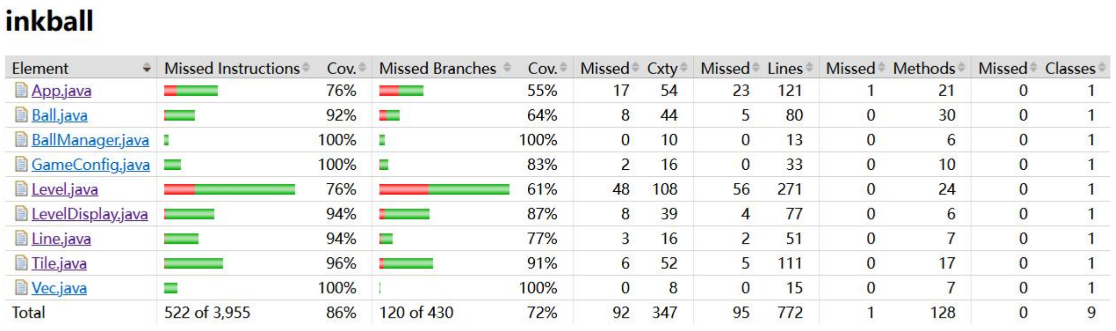
The design made use of multiple classes to separate out logic into different sections

- App: The main application class for the Inkball game, extending PApplet from Processing. This class handles game initialization, configuration loading, user input, user's score and rendering of game elements.
- Ball: represents a ball in the game with properties such as position, velocity, and color. It handles movement, collision detection, and rendering of the ball.
- BallManager: manages the list of all ball of the level. It allows adding, removing, and retrieving colors from the list.
- GameConfig: handles the configuration settings for the game. It loads level data and score modifiers from a specified configuration file.
- Level: represents each game level in the Inkball game. It manages the layout, score, time, balls, lines, and interactions within the level.
- LevelDisplay: manages the loading and representation of a game level. It handles the initial layout of tiles and spawns defined in a specified layout file. It is a separate class for expandability. If you want to add more types of tiles in the future, you can easily get different types of tiles directly from here.
- Line: represents a line made up of a series of points. It provides methods for adding points, drawing the line, detecting collisions with balls, reflecting balls off the line.
- Tile: represents a tile on the game board. It handles tile properties such as position, type, and interactions with balls.
- Vec: represents a 2D vector with x and y components. It provides methods for vector operations such as distance calculation, addition, normalization, and dot product. It is a separate class for expandability. If you want to add more types of collision in the future, you can easily add the calculation methods here.

Implementation of Extension Feature

I choose to implement the Bricks extension. The wall will be cracked when it is hit successfully by once. After it is hit three times, the wall will disappear, and the original place will be shown as a tile. The definition of hitting successfully is the wall is hit by a ball of the corresponding colour, unless the wall is grey or the ball is grey. I used a hitTile function to implement this, in the Tile.java. I used a hit counter to record the times of hitting a wall. When the wall is first hit, the wall will be shown as a cracked condition, when it is hit three times, the wall will disappear, and the type of the wall will change to "none", meaning no wall.

Testcases Code Coverage



inkball

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
App.java		76%		55%	17	54	23	121	1	21	0	1
Ball.java		92%		64%	8	44	5	80	0	30	0	1
BallManager.java		100%		100%	0	10	0	13	0	6	0	1
GameConfig.java		100%		83%	2	16	0	33	0	10	0	1
Level.java		76%		61%	48	108	56	271	0	24	0	1
LevelDisplay.java		94%		87%	8	39	4	77	0	6	0	1
Line.java		94%		77%	3	16	2	51	0	7	0	1
Tile.java		96%		91%	6	52	5	111	0	17	0	1
Vec.java		100%		100%	0	8	0	15	0	7	0	1
Total	522 of 3,955	86%	120 of 430	72%	92	347	95	772	1	128	0	9