**Preliminary Specification — Water Color Sort Puzzle**

Names: Susanna Tang, Ashley Chen, Kayla Chen

**Overall Description:**

Players start a level with either 5, 7, or 9 test tubes randomly filled with different colors.

The objective of this game is to complete all 12 levels — each having various numbers of tubes with random colors inside — by sorting the colors in the tubes accordingly so that each tube is either empty or only filled with one color. A player is able to restart the game at any point.

The general rules of the game are as follows:

1. Players can only move one color from one test tube to the other if the colors on top are the same.
2. A test tube can only be poured into another if the other tube is not at the maximum capacity of four color blocks
3. A complete or empty test tube cannot be selected.
4. Players must proceed level by level, starting from level 1, and cannot skip levels or go back levels.
5. A level is complete when all the test tubes are either empty or completely filled with the same color.
6. A player can restart at any time during a game by pressing the “restart” button.
7. Additional mystery levels only display the top color of the test tube; once revealed, a color will stay revealed.

Players use a mouse to select a test tube. If an unselected tube is clicked, then the tube is selected. If a selected tube is clicked again, the tube is deselected. A selected tube is raised and a deselected tube is unraised. In order to pour a color into another test tube, the player must click on a test tube and click on another test tube to pour the top color. To complete a level, the colors must be sorted correctly. A “congratulations” banner and a “next” button will then appear to prompt the player to advance to the next level. There is a “restart” button at the top left of the game screen. If it is clicked at any time during the game, the game will restart.

Water Color Sort Puzzle does not generate random levels based on an algorithm — the twelve levels are manually made.

**Class/Interface Overview**

**Top Level Class**

**Main** — the main class of the application. Creates the GUI and runs the game’s animation and algorithm. This includes the JFrame, the test tubes, buttons (restart and next), as well as the finished page after a level is complete. Creates animation elements utilized in the game such as pouring from one test tube to another. Detects mouse selection and deselection that occurs in the game such as raising and un-raising the test tube.

**Main Classes**

**Tube** — This class constructs each tube. Each tube is a stack of Colors. Each slot in the stack is a color. The bottom of the stack is the bottom of the tube, and the top of the stack is the top of the tube. Stacks are popped, peeked, and pushed to determine the pouring from one test tube to another. the Stack’s peek function to access the top color. We use the Stack’s pop and push function to pour from one tube to another. Checks when a test tube is complete or not. Gets coordinates of test tubes in a current level for determining selection or deselection. Returns whether a tube is selected or not. Draws tubes using minor class TubeShape. Draws color blocks in appropriate positions for the test tubes. Returns current number of color blocks. Increments color blocks appropriately.

**Level** — Represents one level in the game. Draws 5, 7, or 9 tubes depending on which level it is. Returns the number of tubes, the number of colors in the level, the number of complete tubes, whether the level is a mystery level, and whether or not the level is complete or not. Appropriately increments the number of complete tubes.

**AllLevels** — This class constructs all the levels of the whole game. It will be an array of Levels. We create this object in Main to initialize all levels. It gets the number of levels and completed levels, and sets levels to complete appropriately.

**MakeLevel** — Creates all of the Levels. Each level is an array of tubes. Each tube is a Stack of Colors. We call MakeLevel in Main to create all of the Levels. Levels vary by the number of tubes shown (5, 7, or 9), as well as the color block order. Mystery levels are also created.

**Minor Classes**

**TubeShape** — Extends the Path2D.Float class that stores a geometric path using coordinates. This class constructs the shape of the tube, which is a rectangle with lower two corners rounded. It uses Path2D.Float moveTo, lineTo, and curveTo methods to construct the outline of the tube.

**ColorBlock —** Extends the Color class. This class creates the blocks of colors that are in each test tube. Each ColorBlock has a color and a variable that determines whether the color is a mystery color or not.

**GUI Classes**

See top level class description.

**Rough Class Diagram**

****

**Structural Design**

The following data structures will be used.

| **Description** | **Structure** |
| --- | --- |
| Each Tube | Stack<ColorBlock> |
| Each Level | Tube[] |
| All Levels | Level[] |

**Data Structure Rationale**

We decided to represent each tube with a Stack of ColorBlocks because we only need to access the top color of each tube. When a tube is poured into another tube, only the top color is removed from the current tube and added to the top of another tube. With a Stack, we can use the peek function to access the top color, the pop function to remove the top color, and the push function to add a color to the top of a tube. To display all colors of the tube, we made a temporary Stack to access all colors. This is efficient because each stack has a capacity of four elements which is not time consuming to loop through.

Each level is represented with an array of Tubes. There is no specific order that we need the tubes to be in, and we want to be able to access all tubes and know the index of the current tube at all times, so an array would be the best data structure to represent each level. Plus, we never need to add or remove tubes from each level, so an ArrayList would not be necessary, as it takes up more memory than arrays anyways.

Finally, we represented all levels with an array of levels. We only need to loop through all the levels from level 1 to level 12 (index 0 to index 11). Because of the sequential order of our accessing of each level, an array or linked list would be the most efficient data structure. We chose array over linked list because arrays take up less space than linked lists. We also never need to add or remove levels from our array, so an ArrayList would not be necessary.

**High Level Major Class Specifications**

**Tube**

* Attributes
  + Stack<ColorBlock> colors
  + int currNumBlocks
  + boolean isMystery
  + boolean isSelected
  + int y
  + int originalY
  + int tubeX
  + int tubeY
* Methods
  + void drawTube(Graphics g, int x, int y1)
  + void drawColor(Graphics g, ColorBlock c, int pos, int x, int y)
  + void select()
  + void deselect()
  + boolean getIsSelected()
  + void setTubeX(int x)
  + void setTubeY(int y)
  + void setY(int y)
  + int getY()
  + int getTubeX()
  + int getTubeY()
  + int getOriginalY()
  + Stack<ColorBlock> getColors()
  + boolean isEmpty()
  + boolean isFull()
  + int numBlocks()
  + void addToNumBlocks(int n)
  + boolean isComplete()
  + Color topColor()
  + void pourTo(Tube otherTube)
  + boolean sameColor(Color otherColor)
  + boolean isMystery()

**Level**

* Attributes
  + Tube[] tubes
  + int numTubes
  + int numCompleteTubes
  + boolean isMystery
  + int numColors
* Methods
  + void drawTubes(Graphics g)
  + Tube[] getTubes()
  + int getNumTubes()
  + int getNumColors()
  + int getNumCompleteTubes()
  + void incrementNumCompleteTubes()
  + boolean isComplete()
  + boolean isMystery()

**AllLevels**

* Attributes
  + Level[] levels
  + int numLevels
  + int levelsCompleted
* Methods
  + Level[] getLevels()
  + int getNumLevels()
  + int getNumLevels()
  + int getLevelsCompleted()
  + void setLevelsCompleted(int n)
  + boolean allLevelsCompleted()

**MakeLevel**

* Methods
  + Level level1()
  + Level level2()
  + Level level3()
  + Level level4()
  + Level level5()
  + Level level6()
  + Level level7()
  + Level level8()
  + Level level9()
  + Level level10()
  + Level level11()
  + Level level12()