Bluegrass Community and Technical College

**Programming Requirements Document**

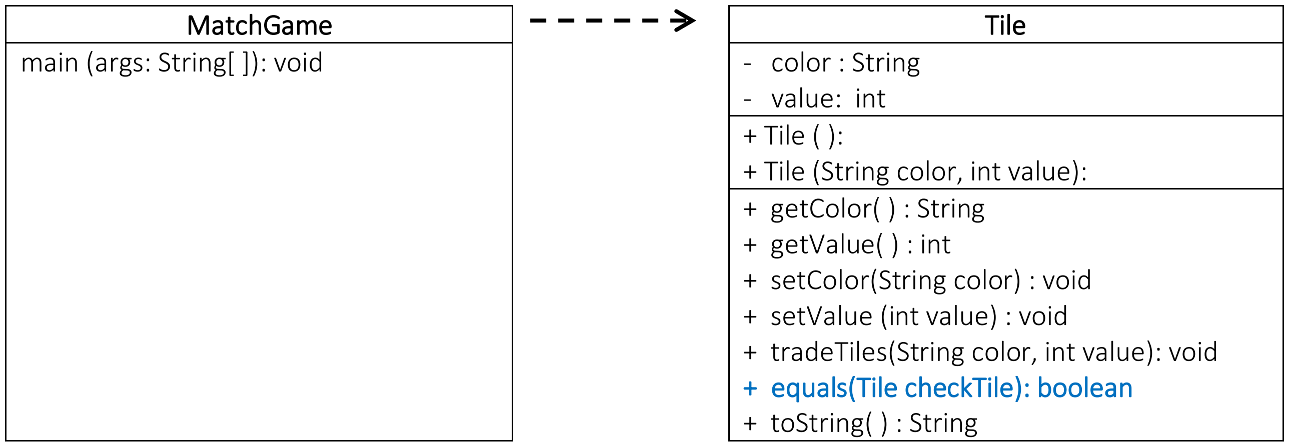
# Tile Game – Take 2

## NARRATIVE DESCRIPTION

Last week we began our study of user-defined Java classes. You were given a program (called a driver) which used a **Tile** class. You learned how to create a new class in Java by coding the **Tile** class. This included defining data for the class (color and value), creating two constructors, two accessors, two mutators, an additional method named tradeTiles(), and overriding the toString() method (from the Object class). The **Tile** class, now that it is created, can be used by any number of games which use those type of tiles.

This week we will reverse your role. You will use your **Tile** class from last week, with an additional modification, and write a driver to create a simple game for toddlers to play. The game will be called **MatchGame**.

This is the new UML class diagram for your latest application:



| MatchGame |
|---|
| main (args: String[ ]): void |

| Tile |
|---|
| - color : String<br>- value: int |
| + Tile ( ):<br>+ Tile (String color, int value): |
| + getColor( ) : String<br>+ getValue( ) : int<br>+ setColor(String color) : void<br>+ setValue (int value) : void<br>+ tradeTiles(String color, int value): void<br>+ equals(Tile checkTile): boolean<br>+ toString( ) : String |

( + represents the public modifier and – represents the private modifier )

### Match Game:

You will create the beginning of a child's game.

Remember: Our tiles contains 104 tiles numbered from 1 to 13 in four different colors (red, green, yellow and blue). There are duplicates (two each) of these tiles.

You will create a portion of a new child's game. In this game, each player will draw 3 tiles. The first person to draw 3 tiles with matching numbers (such as three 7s, or three 10s, etc. ) wins. The goal is for a toddler to recognize numbers.

### Goals:

- Instantiate 3 Tile Objects

  You will simulate a player drawing 3 tiles from a bag of tiles:

  - To draw a single tile:
    - Generate a random integer in the range 1-13 to represent the tile value.
    - Generate a random integer in the range 1-4 to represent a color. (We can later use 1 to mean red, 2 to mean green, 3 to mean yellow, and 4 to mean blue.)

  - We want to create a Tile object (instantiate a Tile object). However, **the constructor for Tile objects requires the color to be a string and not an integer.**

    Use a selection statement to create a new String variable named colorText. In other words, the random integer you generated for the color can be used to give colorText its correct value (Red, Green, Yellow, or Blue). You could use an if structure or a switch structure to do this. Note: You have an integer in the range 1-4 for color and then you are setting colorText to "Red", "Green", "Yellow" or "Blue" based upon the value of the integer.
    - Finally, use the String for color and the integer for value as parameters to the Tile constructor to instantiate a Tile object.

  - Generate two more Tiles like the first.

    NOTE: We have not learned about arrays yet. No arrays are allowed in this solution.

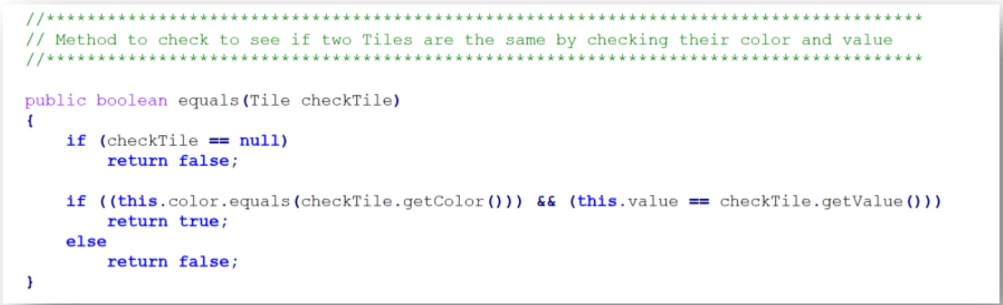
- Play a Match Game

  Now that 3 Tile objects have been instantiated, use the Tile objects and Tile methods (accessors, toString() etc.) when you need data about a tile and DO NOT use the random number integers or strings. This is an exercise in using objects.

  In our set of tiles, we have 2 of every color/number combination. For example, there are two Red 8s, two Blue 10s, etc.

  Since we are generating 3 Tile objects randomly, it is possible that all 3 tiles you generated are identical (for example, 3 Blue 8s). This is invalid because the set of tiles do not have 3 identical tiles. We need to check for this and then generate a 3rd new tile if all 3 are identical.

  - To do this, you will need a new method added to your Tile class:



```java
//**************************************************************************************
// Method to check to see if two Tiles are the same by checking their color and value
//**************************************************************************************

public boolean equals(Tile checkTile)
{
    if (checkTile == null)
        return false;

    if ((this.color.equals(checkTile.getColor())) && (this.value == checkTile.getValue()))
        return true;
    else
        return false;
}
```

Save the equals() method in Tile.

From inside your driver, you can use the equals() method can be used to see if two Tiles are identical. For example, if you have Tile objects named **randomTile1** and **randomTile2**, the following would return **true** if the two tiles are identical

```
    randomTile1.equals(randomTile2)
```

Such a condition could be used in an if statement or while statement.

- You can write a compound condition to see if the first tile is equal to the second tile and the second tile is equal to the third tile (all 3 are equal). Use a while loop to generate a new third tile if all three are equal. You need a loop rather than an if statement because you could generate the third tile again equal to the first two. You want to repeat the loop as long as the 3 are equal. Don't forget there is a **tradeTiles()** method in the Tile class which can be used to get a new Tile.

- After all 3 tiles are created and they are not all the same, display the 3 tiles (color and value). This lets the user see his/her tiles. You can use your toString() methods to display the tiles.

- If all 3 are the same number (color doesn't matter), tell the player they "WON" the game. You can use the Tile accessors to retrieve the value of each tile.

- Finally, create a loop to generate another set of Tiles if the user wants to play the game again. Allow the user to play this as long as they like (until they enter quit or something similar).

Remember: This is just a portion of a game. In a real scenario, you would have other players, etc.

## New Concepts Assessed and Illustrated (in addition to any previously learned)

- Instantiating objects from a user-defined class
- Using accessors to retrieve object data
- Adding and using an equals() method to compare two objects
- Using a toString( ) method to display class data

## Software Requirements

Standard Requirements listed in previous weeks are implicit requirements now (documentation, free of compile errors, user interface is easy, etc.).

R1: Create three Tile objects. Make sure all 3 are not the same.
R2: Create a working equals() method for the Tile class.
R3: Tile objects are clearly displayed.
R4: The user interface is easy to use.
R5: The user is told if they won or lost the game
R6: The game can be played as many times as the user wishes.
R7: Class methods are used correctly.

## Security Considerations

The public and private access modifiers have security implications. Use a private modifier for all instance data items. Create accessors for instance data items only when absolutely needed.  Do not include sensitive or private information in data returned by the toString() method.