Programming Assignment 3
Search The Picture

1 Objectives

1. Getting Introduced to Multi-Dimensional Arrays in JAVA.
2. Becoming familiar with problem thinking using appropriate data structure.
3. Becoming familiar with JUnit Tests.

2 Problem Statement

Three weeks ago, MyOwnBusiness Inc. received an urgent call from the IIHF (International Ice Hockey Federation) requesting a system to raise an alarm to the referee when there are too many players from the same team inside the rink. The system will be composed of three parts:

- A digital camera in the ceiling to take photos of the rink every second.
- A software module to extract the position of each player from the photo taken by the digital camera.
- A software module to count the number of players from the same team inside the hockey rink.

The team has just finished the module to count the number of players inside the hockey rink, so now it is time to implement the module to extract the players' positions from the photo taken by the digital camera.

The **photo** taken by the camera is given as a String[], where the x-th character of the y-th element is the color of the 2 x 2 square whose upper-left corner is at **(2*x, 2*y)**. Colors are either uppercase letters ('A'-'Z') or digits ('0'-'9'). Uppercase letters represent terrain features (floor, chairs, spectators, etc.) and each digit X represents the color of the uniform used by the X-th team.

Two squares A and B belong to the same object if and only if there exists a chain of squares where the first square is A, the last square is B, each pair of consecutive squares in the chain shares a common edge and all the squares in the chain have the same color. The position of an object C is the center of its bounding box, where its bounding box is defined as smallest axis-aligned box that contains all the object's squares. An object's **area** is defined as the sum of the areas of all the squares that compose the object. An object is a player from the i-th team if and only if it is colored with the digit i and its area is at least **threshold**.

Return a java.awt.Point[] containing all the players in the photo from the **k**-th team. Each element should represent a single player and be formatted **java.awt.Point(X,Y)**, where (X, Y) is the **center** of the player's bounding box, and X and Y have **no extra leading zeros**. **Sort** the players in increasing order by x-coordinate. Sort players with the same x-coordinate in increasing order by y-coordinate.

3 Integration

Organize your code under package with name

`eg.edu.alexu.csd.datastructure.iceHockey.cs<your-two-digits-class-number>`

and you need to implement the following interface

```java
package eg.edu.alexu.csd.datastructure.iceHockey;
public interface IPlayersFinder {

    /**
     * Search for players locations at the given photo
     * @param photo
     *     Two dimension array of photo contents
     *     Will contain between 1 and 50 elements, inclusive
     * @param team
     *     Identifier of the team
     * @param threshold
     *     Minimum area for an element
     *     Will be between 1 and 10000, inclusive
     * @return
     *     Array of players locations of the given team
     */
    java.awt.Point[] findPlayers(String[] photo, int team, int threshold);
}
```

4 Constraints

- **photo** will contain between 1 and 50 elements, inclusive.
- Each element of **photo** will contain between 1 and 50 elements, inclusive.
- Each element of **photo** will contain the same number of characters.
- Each element of **photo** will contain only uppercase letters ('A'-'Z') and digits ('0'-'9').

- **k** will be between 0 and 9, inclusive.
- **threshold** will be between 1 and 10000, inclusive.
- You should handle empty/null image.

5 Examples

1.
   ```
findPlayers(
  {
   "33JUBU33",
   "3U3O4433",
   "O33P44NB",
   "PO3NSDP3",
   "VNDSD333",
   "OINFD33X"
  },  3,  16);
```
   Returns: { (4,5), (13,9), (14,2) }
   There are four groups of adjacent pixels with color '3'. However, the first one is too small to be considered (its area is 12, which is smaller than the threshold).

2.
   ```
findPlayers(
  {
   "44444H44S4",
   "K444K4L444",
   "4LJ44T44XH",
   "444O4VIF44",
   "44C4D4U444",
   "4V4Y4KB4M4",
   "G4W4HP4O4W",
   "4444ZDQ4S4",
   "4BR4Y4A444",
   "4G4V4T4444"
  },  4, 16);
```
   Returns: { (3,8), (4,16), (5,4), (16,3), (16,17), (17,9) }

3.
```
findPlayers(
  {
    "8D88888J8L8E888",
    "88NKMG8N8E8JI88",
    "888NS8EU88HN8EO",
    "LUQ888A8TH8OIH8",
    "888QJ88R8SG88TY",
    "88ZQV88B88OUZ8O",
    "FQ88WF8Q8GG88B8",
    "8S888HGSB8FT8S8",
    "8MX88D88888T8K8",
    "8S8A88MGVDG8XK8",
    "M88S8B8I8M88J8N",
    "8W88X88ZT8KA8I8",
    "88SQGB8I8J88W88",
    "U88H8NI8CZB88B8",
    "8PK8H8T8888TQR8"
  }, 8, 9);
);
```

Returns:
{ (1, 17), (3, 3), (3, 10), (3, 25), (5, 21), (8, 17), (9, 2), (10,9), (12,23), (17,16), (18,3), (18,11), (18,28), (22,20), (23,26), (24,15), (27,2), (28,26), (29,16) }

4.
```
findPlayers(
  {
    "11111",
    "1AAA1",
    "1A1A1",
    "1AAA1",
    "11111"
  }, 1, 3
);
```

Returns: { (5,5), (5,5) }

6 Deliverables

- Develop this assignment in Java language.
- You should follow the given interface specified in the Definition section.
- You should push your code to the repository.
- You should try out your code using http://onlinetester.tk/
- Test your implementation using JUnit Test. Your JUnit tests **will be graded** based on coverage for the different cases.
- You should integrate Checkstyle library http://checkstyle.sourceforge.net/ with your IDE to ensure that your code style follows the JAVA coding style standards.
- You should work individually.
- Late submission is accepted for only one week.
- Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.
- The problem is based on TopCoder Inc. Single Round Match (SRM) problem.

Good Luck :)