Problem Description:

In this project, we are asked to make a game with the scenario mentioned below.
You will develop a two-player board game. The board contains 16 pieces that are
organized in a 4*4 square. Each piece can be either black or white, tall or short, square or round, and hollow or solid. The game starts with an empty board.
Players take turns in playing. At each round, a player picks a piece and gives it to the other player, who then places it on the board. A player wins the game after a move if there are four pieces that share one property horizontally, vertically, or diagonally. You can name each piece with the four properties such that ▫BTSH▫ means a black, tall, square, and hollow. And we make sure there is a space between them. So while all strings is being divided 4 lines, even if each lines don't have same amount of strings, we must have 4 rows as shown on the assignment. In addition, we ask the user if she wants to start a new game or

if she wants to continue an ongoing game. Project also want to avoid using an input from us again and not to use a used position once more.

Problem Solution:

Since our inputs are 4 digits, I set my basic board to 4 strings + space + 4 strings, so when I change the board, I do not break the main board.
I used the String change(); method which accepts two trings and one integer to change the board on each input. In this method I fixed the string up to the position
I would like to change, then I printed out the input and added the rest of it.I set string to change by multiplying each x1 for user, x2 for program position by 20 and multiplying y1 for user, y2 for program by 5. Because each line consists of 20 spaces and 5 spaces between each two board elements.I used two methods
that accept a String named userBW (); and a string called progBW (); in order to be able to ask or get what the input of the program or user is. I sent an empty string to these methods, and at the end of each question I added the answers to the empty string. I used rand.nextInt (2) in the progBW method to get the input randomly assigned by the computer, receiving 0 and 1 status, a reply for if (0), and a reply for elseif (1). To avoid using an used input again and not to use a used position once more, first, I write the boolean controlSame (); method, which accepts two strings, to check if the input was already used. I provide such
a situation by returning false. i is increased by five, look between each i and (i + 4) and enter again if input equals. Using boolean sameChanging(); method, we know if the position entered is full. If there is an 'E' at the position entered, we know that it is empty, and we can change the string by inserting it with
true inside while. Otherwise, I'd like to write another input. I do these operations in the while loop, which allows you to change the whole string to keep the
game up to either win or tie. To do this, I use boolean important(); method. Because the winning conditions are diagonally, horizontally and vertically, I have
returned the if statement in the for loop method in each conditional boolean method. After each position is entered to understand who has won the game, I write the
if (important (blanks) == true) method and I will stop the game with break and know who won it. Finally, I used two methods to generate the output as desired by the
project. I removed the excess gaps in the String arrangeLine(); method and reduced it to a single gap. I've added an empty string with the if statements using For loop.
I used the spaceCount(); method to subdivide every fourth space. At last, I ask "Do you wanna play new game?", if statement check answer and if answer is yes, whole game(); method starts from the beginning, otherwise continue an ongoing

```
game.

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Random;
import java.util.Scanner;

public class saddas {

	public static void main(String[] args) throws FileNotFoundException {
		game();
	}

	public static void game() throws FileNotFoundException {
		System.out.println("This is a board game and the game's rule is:");
		System.out.println(	"At each round, a player picks a piece and
gives it to the other player, who then places it on the board.");
		System.out.println("To win the game, after a move if there should be
four pieces that share one property horizontally, vertically, or diagonally.");
		System.out.println();
		Scanner console = new Scanner(System.in);
		Scanner input = new Scanner(new
File("C:\\Users\\Hp\\Desktop\\input.txt"));		// input represents
input.text.
		String text = input.nextLine();
							// text represents string
form of input.
		String blanks = "";
				// blanks will represent text which I will desire.
		for (int i = 0; i < text.length(); i += 2) {
	// This for loop provide space between 'E' to be four space. Make 80
character-string.
			blanks += text.substring(i, i + 1) + "    ";

		}
		Random rand = new Random();
		String first = arrangeLine(blanks);
		// first represents empty board like an Example 1.
		spaceCount(first);
		while (!important(blanks)) {
		// This while statement allows to return program the desired level.
			String x = progBW("");
			// x represents program's input.

			while (!controlSame(blanks, x)) {
		// This while statement check blanks if the x was already used.
				x = progBW("");

			}
			System.out.println();
			System.out.println("Program picks: " + x);
			System.out.println("Please choose coordinate:");
			int x1 = console.nextInt();
			// x1 represents user's coordinate which is chosen.
			int y1 = console.nextInt();
			// y1 represents user's coordinate which is chosen.
			int changingUser = x1 * 20 + y1 * 5;
		// changingUser represents user's coordinate in the form of String.

			while (!sameChanging(blanks, changingUser)) {
	// This while statement check blanks if the position entered is full.
				System.out.println("Writing wrong position. Write
again:");
				x1 = console.nextInt();
```

```
                        y1 = console.nextInt();
                        changingUser = x1 * 20 + y1 * 5;

                  }

                  blanks = change(blanks, x, changingUser);
      // By this statement, blanks change according to input and coordinate.
                  String blanks1 = arrangeLine(blanks);
            // blanks1 is the form of blanks with space.
                  System.out.println();
                  spaceCount(blanks1);

                  if (important(blanks) == true) {
            // This if statement check user's win-lose situation if user wins.
                        System.out.println();
                        System.out.println("You have won.");
                        break;
                  }
                  System.out.println();
                  String y = userBW("");
      // y represents user's input.

                  while (!controlSame(blanks, y)) {                           //
This while statement check blanks if the y was already used.
                        System.out.println();
                        System.out.println("You write wrong input. Write
again.");
                        System.out.println();
                        y = userBW("");

                  }
                  System.out.println(y);
                  int x2 = rand.nextInt(4);
      // x2 represents program's coordinate which is randomly chosen.
                  int y2 = rand.nextInt(4);
      // y2 represents program's coordinate which is randomly chosen.
                  int changingProg = x2 * 20 + y2 * 5;                        //
changingProg represents program's coordinate in the form of String.

                  while (!sameChanging(blanks, changingProg)) {        // This
while statement checks blanks if the position entered is full.
                        x2 = rand.nextInt(4);
                        y2 = rand.nextInt(4);
                        changingProg = x2 * 20 + y2 * 5;

                  }
                  System.out.println("Program coordinate is " + x2 + " " + y2);
                  blanks = change(blanks, y, changingProg);            // By
this statement, blanks change according to input and coordinate.
                  String blanks2 = arrangeLine(blanks);                       //
blanks1 is the form of blanks with space.
                  System.out.println();
                  spaceCount(blanks2);
                  if (important(blanks) == true) {                            //
This if statement checks program's win-lose situation if program wins.
                        System.out.println();
                        System.out.println("You have lost.");
                        break;
                  }
                  if (draw(blanks) == true) {
      // This if statement checks draw situation.
                        System.out.println();
                        System.out.println("Draw.");
                  }
```

```
                System.out.println();
                System.out.println("Do you wanna play new game?, if your
answer is no, you continue an ongoing name.");
                String yesNo = console.next();
      // yesNo represents user's answer.

                if (yesNo.equalsIgnoreCase("yes")) {                        //
This if else statement checks the answer yes or no.
                      game();
                }
           }

      }

      public static String userBW(String config) {                     // This
method enables user to write input as the desired.
           Scanner console = new Scanner(System.in);
           System.out.println("Black or White? For Black and White write 'B'
and 'W' respectively.");
           String answer1 = console.nextLine();                              //
answer1 represent answer of 'Black or White?'
           if (answer1.equalsIgnoreCase("B")) {                              //
This if else statement checks answer1 according to 'B' and 'W'.
                config += 'B';
           } else if (answer1.equalsIgnoreCase("W")) {
                config += 'W';
           }
           System.out.println("Tall or Short? For Tall and Short write 'T' and
'S' respectively.");
           String answer2 = console.nextLine();                              //
answer2 represent answer of 'Tall or Short?'
           if (answer2.equalsIgnoreCase("T")) {                              //
This if else statement checks answer2 according to 'T' and 'S'.
                config += 'T';
           } else if (answer2.equalsIgnoreCase("S")) {
                config += 'S';
           }
           System.out.println("Square or Round? For Square and Round write 'S'
and 'R' respectively.");
           String answer3 = console.nextLine();                              //
answer3 represent answer of 'Square or Round?'
           if (answer3.equalsIgnoreCase("S")) {                              //
This if else statement checks answer3 according to 'S' and 'R'.
                config += 'S';
           } else if (answer3.equalsIgnoreCase("R")) {
                config += 'R';
           }
           System.out.println("Hollow or Solid? For Hollow and Solid write 'H'
and 'S' respectively.");
           String answer4 = console.nextLine();                              //
answer4 represent answer of 'Hollow or Solid?'
           if (answer4.equalsIgnoreCase("H")) {                              //
This if else statement checks answer4 according to 'H' and 'S'.
                config += 'H';
           } else if (answer4.equalsIgnoreCase("S")) {
                config += 'S';
           }
           return config;
      }

      public static String progBW(String config) {                     // This
method enables program to write input randomly as the desired.
           Random rand = new Random();
           int bw = rand.nextInt(2);
```

```
      // bw represents 'B' and 'W'.
            if (bw == 1) {
            // This if else statement checks bw.
                  config += 'B';
            } else {
                  config += 'W';
            }
            int ts = rand.nextInt(2);
      // ts represents 'T' and 'S'.
            if (ts == 1) {
            // This if else statement checks ts.
                  config += 'T';
            } else {
                  config += 'S';
            }
            int sr = rand.nextInt(2);
      // sr represents 'S' and 'R'.
            if (sr == 1) {
            // This if else statement checks sr.
                  config += 'S';
            } else {
                  config += 'R';
            }
            int hs = rand.nextInt(2);
      // hs represents 'H' and 'S'.
            if (hs == 1) {
            // This if else statement checks hs.
                  config += 'H';
            } else {
                  config += 'S';
            }

            return config;

      }

      public static String change(String newConfiguration, String character, int
changing1) {                  // By this method, blanks change according to
input and coordinate.
            String newCBoard = newConfiguration.substring(0, changing1) +
character
                        + newConfiguration.substring(changing1 + 4);
            return newCBoard;
      }

      public static String arrangeLine(String sadads) {                // 80-
character string will be shape of that only space between each element by using
this method.
            String arrange = "";
      // By using arrange variable, it allows me to write string as a space
between each position.
            for (int j = 0; j < sadads.length() - 2; j++) {            // This
for loop allows me to look at all the indexes from zero to sadads.length()-2.
                  if (sadads.charAt(j) == 'E') {
      // This if else statement ensures that there is a space between each
position.
                        arrange += sadads.charAt(j) + " ";
                  } else if (sadads.charAt(j) != ' ' && sadads.charAt(j + 1) ==
' ' && sadads.charAt(j + 2) != ' ') {
                        arrange += sadads.charAt(j) + " ";

                  } else if (sadads.charAt(j) != ' ' && sadads.charAt(j + 1) !=
' ') {
                        arrange += sadads.charAt(j);
```

```
                }

            }
            arrange += sadads.charAt(sadads.length() - 2);
            return arrange;

      }

      public static void spaceCount(String space) {                    // This
method provides to divide 4 lines as the desired.
            int v1 = 0;                                                 // v1
represents the index at which the fourth space is located.
            int v2 = 0;                                                 // v2
represents the index at which the 8th space is located.
            int v3 = 0;                                                 // v3
represents the index at which the 12th space is located.
            int count = 0;                                              // count
represents the number of spaces.
            for (int i = 0; i < space.length() - 2; i++) {              // This
for loop provides to look all indexes of string.
                  if (space.charAt(i) != ' ' && space.charAt(i + 1) == ' ' &&
space.charAt(i + 2) != ' ') {      //This if statement enables to count the
number of spaces.
                        count++;

                  }
                  if (count == 4 && space.charAt(i) == ' ') {
                  // This if else statement provides line-by-line division
according to the number of spaces.
                        v1 = i + 1;
                        System.out.println(space.substring(0, v1));
                  } else if (count == 8 && space.charAt(i) == ' ') {
                        v2 = i + 1;
                        System.out.println(space.substring(v1, v2));
                  } else if (count == 12 && space.charAt(i) == ' ') {
                        v3 = i + 1;
                        System.out.println(space.substring(v2, v3));
                  }

            }
            System.out.println(space.substring(v3));
      }

      public static boolean important(String newConfiguration) {             //
This method checks condition of game to end.
            for (int i = 0; i < 4; i++) {
                  for (int j = 0; j < 5; j++) {
                        int k = i * 20;
                        if ((newConfiguration.charAt(k + j) ==
newConfiguration.charAt(k + 5 + j))            // This if statement checks
horizontally ending.
                                    && (newConfiguration.charAt(k + 10 + j) ==
newConfiguration.charAt(k + 15 + j))
                                    && (newConfiguration.charAt(k + 5 + j) ==
newConfiguration.charAt(k + 10 + j))
                                    && (newConfiguration.charAt(k + j) != 'E' &&
newConfiguration.charAt(k + j) != ' ')) {
                              return true;
                        }

                  }
            }
            for (int j = 0; j < 4; j++) {
                  for (int i = 0; i < 5; i++) {
```

```
                        int l = j * 5;
                        if ((newConfiguration.charAt(l + i) ==
newConfiguration.charAt(l + 20 + i))                // This if statement checks
vertically ending.
                                    && (newConfiguration.charAt(l + 40 + i) ==
newConfiguration.charAt(l + 60 + i))
                                    && (newConfiguration.charAt(l + 20 + i) ==
newConfiguration.charAt(l + 40 + i))
                                    && (newConfiguration.charAt(l + i) != 'E' &&
newConfiguration.charAt(l + i) != ' ')) {
                              return true;
                        }
                  }
            }
            for (int j = 0; j < 4; j++) {
                  if ((newConfiguration.charAt(0 + j) ==
newConfiguration.charAt(25 + j))                          // This if statement
checks diagonally ending.
                              && (newConfiguration.charAt(50 + j) ==
newConfiguration.charAt(75 + j))
                              && (newConfiguration.charAt(25 + j) ==
newConfiguration.charAt(50 + j))
                              && (newConfiguration.charAt(j) != 'E' &&
newConfiguration.charAt(j) != ' ')) {
                        return true;
                  }
            }
            for (int j = 0; j < 4; j++) {
                  if ((newConfiguration.charAt(15 + j) ==
newConfiguration.charAt(30 + j))                    // This if statement checks
back diagonally ending.
                              && (newConfiguration.charAt(45 + j) ==
newConfiguration.charAt(60 + j))
                              && (newConfiguration.charAt(30 + j) ==
newConfiguration.charAt(45 + j))
                              && (newConfiguration.charAt(j) != 'E' &&
newConfiguration.charAt(j) != ' ')
                              && (newConfiguration.charAt(j + 15) != 'E' &&
newConfiguration.charAt(j + 15) != ' ')) {
                        return true;
                  }
            }

            return false;
      }

      public static boolean controlSame(String sameConfig, String xSame) {
            // This method enables to avoid using an used input again.

            for (int i = 0; i < sameConfig.length(); i += 5) {
                        // To do this, this if statement checks all position to
look if there is same input.
                  if (sameConfig.substring(i, i + 4).equals(xSame)) {

                        return false;
                  }
            }

            return true;
      }

      public static boolean sameChanging(String sameConfig, int sameChanging) {
            // This method enables not to use a used position once more
            if (sameConfig.charAt(sameChanging) == 'E') {
```

```
                        // To do this, this if statement checks 'E' at the
position which is chosen.
                  return true;
            }

            return false;
      }

      public static boolean draw(String drawConfig) {                  // This
method checks draw situation.
            for (int i = 0; i < drawConfig.length(); i += 5) {
                  if (drawConfig.charAt(i) == 'E') {                         //
To do this, this if statement looks all position until the all position is full.
                        return false;
                  }
            }
            return true;
      }

}
```