blastUISpecialGem()

```
public void blastUISpecialGem() {
    this.listOfGems = this.gemBoard.getListOfGems();
    int index = 0;
    for (int k = 0; k < 64; k++) {
       if (this.listOfGems.get(k).getValue() == 6) {
          int x_coordinate = k / 8;
          int y_coordinate = k % 8;
          this.blast_coordinates[index][0] = x_coordinate;
          this.blast_coordinates[index][1] = y_coordinate;
          ++index;
          this.tiles[x_coordinate][y_coordinate].setIcon(liner);
       }
    }
    this.dropSpecialGemsCase();
  }
```

EC:

k=[0,64)

Test Case:

Input:

k=10

Expected output:

Loop Continues

ButtonHandler()

```
private class ButtonHandler implements ActionListener {

   @Override
   public void actionPerformed(ActionEvent ae) {
      LoginFrame.playSound("D:\\FAST\\Semester 5\\Object Oriented Analysis and Design\\Project\\Bonus Part\\sounds\\click.wav");
      Object source = ae.getSource();
      for (int i = 0; i < 8; i++) {
         for (int j = 0; j < 8; j++) {
            if (source == tiles[i][j]) {
               tiles[i][j].setBackground(Color.yellow);
               processClick(i, j);
               return;
            }
         }
      }
   }

}
```

EC:

i=[0,8)

j=[0,8)

Test Case:

Input:

i=4

j=3

Expected output:

Outer and Inner Loop Continues

isValidMove()

```
private boolean isValidMove() {
    if ((this.coordinates[0][0] - 1) >= 0) {
        if ((this.coordinates[0][0] - 1) == this.coordinates[1][0] && this.coordinates[0][1] == this.coordinates[1][1]) {
            return true;
        }
    }
    if ((this.coordinates[0][1] - 1) >= 0) {
        if (this.coordinates[0][0] == this.coordinates[1][0] && (this.coordinates[0][1] - 1) == this.coordinates[1][1]) {
            return true;
        }
    }
    if ((this.coordinates[0][1] + 1) < 8) {
        if (this.coordinates[0][0] == this.coordinates[1][0] && (this.coordinates[0][1] + 1) == this.coordinates[1][1]) {
            return true;
        }
    }
    if ((this.coordinates[0][0] + 1) < 8) {
        if ((this.coordinates[0][0] + 1) == this.coordinates[1][0] && this.coordinates[0][1] == this.coordinates[1][1]) {
            return true;
        }
    }
    return false;
}
```

EC1:

1. (this.coordinates[0][0] - 1) >= 0
2. ((this.coordinates[0][0] - 1) == this.coordinates[1][0] && this.coordinates[0][1] == this.coordinates[1][1])

EC2 :

1. (this.coordinates[0][1] - 1) >= 0
2. (this.coordinates[0][0] == this.coordinates[1][0] && (this.coordinates[0][1] - 1) == this.coordinates[1][1])

EC3 :

1. ((this.coordinates[0][1] + 1) < 8)
2. (this.coordinates[0][0] == this.coordinates[1][0] && (this.coordinates[0][1] + 1) == this.coordinates[1][1])

EC4 :

1. ((this.coordinates[0][0] + 1) < 8)
2. ((this.coordinates[0][0] + 1) == this.coordinates[1][0] && this.coordinates[0][1] == this.coordinates[1][1])

Test Case1:

Input:

this.coordinates[0][0]=6
this.coordinates[1][0]=5
this.coordinates[0][1]=6
this.coordinates[1][1]=6

Expected output:

true

Test Case2:

Input:

this.coordinates[0][0]=6
this.coordinates[1][0]=6
this.coordinates[0][1]=5
this.coordinates[1][1]=6

Expected output:

true

Test Case3:

Input:

this.coordinates[0][0]=5
this.coordinates[1][0]=5
this.coordinates[0][1]=6
this.coordinates[1][1]=7

Expected output:

true

Test Case4:

Input:

this.coordinates[0][0]=4
this.coordinates[1][0]=5
this.coordinates[0][1]=6
this.coordinates[1][1]=6

Expected output:

true

isValidMove()

```
private void processClick(int i, int j) {
    ++this.counter;
    if (this.counter == 1) {
       this.coordinates[0][0] = i;
```

```
            this.coordinates[0][1] = j;
        }
        if (this.counter == 2) {
            this.coordinates[1][0] = i;
            this.coordinates[1][1] = j;
            if (this.isValidMove()) {
                Icon temp = this.tiles[(this.coordinates[0][0])][(this.coordinates[0][1])].getIcon();
                this.tiles[(this.coordinates[0][0])][(this.coordinates[0][1])].setIcon(null);

this.tiles[(this.coordinates[0][0])][(this.coordinates[0][1])].setIcon(this.tiles[(this.coordinates[1][0])][(this.coordinates[1][1])].getIco
n());
                this.tiles[(this.coordinates[1][0])][(this.coordinates[1][1])].setIcon(temp);
                this.gemBoard.swapGem(coordinates);
                int delay = 1000;//specify the delay for the timer
                Timer timer = new Timer(delay, e -> {
                    //The following code will be executed once the delay is reached
                    if(this.gemBoard.findSpecialGemOccurences())
                    {
                        this.blastUISpecialGem();
                        Timer nestedTimer = new Timer(delay, e1 -> {
                            if (this.gemBoard.findOccurences()) {
                                do {
                                    this.blastUIGem();
                                } while (this.gemBoard.findOccurences());
                            }
                        });
                        nestedTimer.setRepeats(false);//make sure the timer only runs once
                        nestedTimer.start();
                    }
                    else if (this.gemBoard.findOccurences()) {
                        do
                        {
                            this.blastUIGem();
                        }while(this.gemBoard.findOccurences());
                                                    }
                                                    else {
                                                            final Icon temp2 =
this.tiles[(this.coordinates[0][0])][(this.coordinates[0][1])].getIcon();

                     this.tiles[(this.coordinates[0][0])][(this.coordinates[0][1])].setIcon(null);

                     this.tiles[(this.coordinates[0][0])][(this.coordinates[0][1])].setIcon(this.tiles[(this.coordinates[1][0])][(this.coordinates[1][1
])].getIcon());

                     this.tiles[(this.coordinates[1][0])][(this.coordinates[1][1])].setIcon(temp2);
                                                            this.gemBoard.swapGem(coordinates);
                                                            LoginFrame.playSound("D:\\FAST\\Semester 5\\Object Oriented
Analysis and Design\\Project\\Bonus Part\\sounds\\no.wav");
                                                    }
                    });
                    timer.setRepeats(false);//make sure the timer only runs once
                    timer.start();
                }
                int delay = 500;//specify the delay for the timer
                Timer timer = new Timer(delay, e -> {
                    this.tiles[(this.coordinates[0][0])][(this.coordinates[0][1])].setBackground(Color.BLACK);
                    this.tiles[(this.coordinates[1][0])][(this.coordinates[1][1])].setBackground(Color.BLACK);
```

```
        });
        timer.setRepeats(false);//make sure the timer only runs once
        timer.start();
        this.counter = 0;
    }
}
```

EC1 :

this.counter==1

EC2 :

this.counter==2

Test Case1:

Input:

this.counter=1

Expected output:

{ this.coordinates[0][0] = i;
this.coordinates[0][1] = j; }

Test Case2:

Input:

this.counter=2

Expected output:

{ this.coordinates[1][0] = i;
this.coordinates[1][1] = j; }