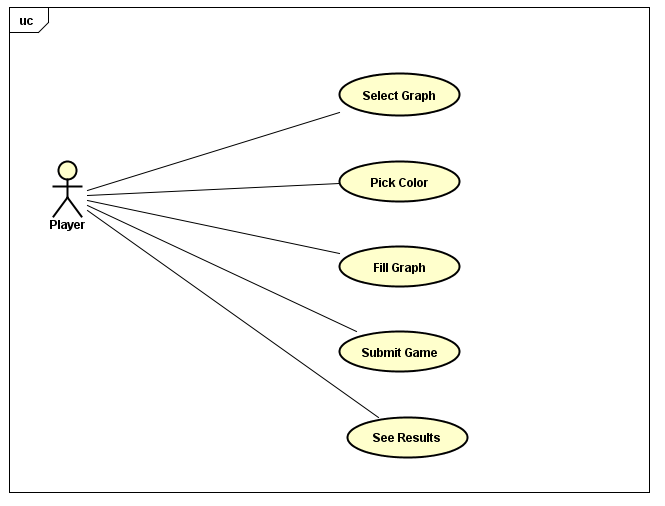
This week I have gone through math concepts of graph coloring to have better knowledge of graph coloring concept. Below I'm enclosing links to content I have gone through

1. <http://www.math.ucsd.edu/~fan/wp/cgame.pdf>

2. <http://www.math.nsysu.edu.tw/~zhu/papers/game/planar.pdf>

3. <http://kam.mff.cuni.cz/kseminar/clanky/Kierstad-asymmetric_graph_games.pdf>

Then I have started with use case diagram for basic graph coloring game with one player though it is not the finalized one will be using this for reference. Further while implementing the project this may change. So here as we have decided to implement pre determined graphs as of now player will be displayed with a set of graphs on interface allowing him to select a graph of his interest. Then on selecting a graph he will be directed to selected graph with a set of colors. Then player will pick a color and fill graph with picked color. Upon completing coloring player will submit game stopping timer. On submission he will be directed to result page.



Further as we have chosen greenfoot for implementing game have gone through methods in green foot for drawing shapes. Some of them are

* public void drawImage([GreenfootImage](https://www.greenfoot.org/files/javadoc/greenfoot/GreenfootImage.html) image,

int x,

int y)

Draws the given Image onto this image

****Parameters:****

image - The image to draw onto this one.

x - x-coordinate for drawing the image.

y - y-coordinate for drawing the image.

* public void drawRect(int x,

int y,

int width,

int height)

Draw the outline of the specified rectangle. The left and right edges of the rectangle are at x and x + width. The top and bottom edges are at y and y + height. The rectangle is drawn using the current color.

**Parameters:**

x - the *x*coordinate of the rectangle to be drawn.

y - the *y*coordinate of the rectangle to be drawn.

width - the width of the rectangle to be drawn.

height - the height of the rectangle to be drawn.

* public void drawShape(java.awt.Shape shape)

Draw a shape directly on the image. Shapes are specified by the [shape interface](http://java.sun.com/javase/6/docs/api/java/awt/Shape.html).

****Parameters:****

shape - the shape to be drawn.

* public void drawPolygon(int[] xPoints,

int[] yPoints,

int nPoints)

Draws a closed polygon defined by arrays of *x* and *y* coordinates. Each pair of (*x*, *y*) coordinates defines a point.

This method draws the polygon defined by nPoint line segments, where the first nPoint - 1 line segments are line segments from (xPoints[i - 1], yPoints[i - 1]) to (xPoints[i], yPoints[i]), for 1 ≤ *i* ≤ nPoints. The figure is automatically closed by drawing a line connecting the final point to the first point, if those points are different.

**Parameters:**

xPoints - an array of x coordinates.

yPoints - an array of y coordinates.

nPoints - the total number of points.

* public void drawLine(int x1,

int y1,

int x2,

int y2)

Draw a line, using the current drawing color, between the points (x1, y1) and (x2, y2).

**Parameters:**

x1 - the first point's *x*coordinate.

y1 - the first point's *y*coordinate.

x2 - the second point's *x*coordinate.

y2 - the second point's *y*coordinate.

* public void drawOval(int x,

int y,

int width,

int height)

Draw an oval bounded by the specified rectangle with the current drawing color.

**Parameters:**

x - the *x*coordinate of the upper left corner of the oval to be drawn.

y - the *y*coordinate of the upper left corner of the oval to be drawn.

width - the width of the oval to be drawn.

height - the height of the oval to be drawn.

**Then to fill Color to shapes there are methods to fill colors**

* public void fillOval(int x,

int y,

int width,

int height)

Fill an oval bounded by the specified rectangle with the current drawing color.

**Parameters:**

x - the *x*coordinate of the upper left corner of the oval to be filled.

y - the *y*coordinate of the upper left corner of the oval to be filled.

* public void fillShape(java.awt.Shape shape)

Draw a filled shape directly on the image. Shapes are specified by the [shape interface](http://java.sun.com/javase/6/docs/api/java/awt/Shape.html).

****Parameters:****

shape - the shape to be filled.

width - the width of the oval to be filled.

height - the height of the oval to be filled.

* public void fillPolygon(int[] xPoints,

int[] yPoints,

int nPoints)

Fill a closed polygon defined by arrays of *x*and *y*coordinates.

This method draws the polygon defined by nPoint line segments, where the first nPoint - 1 line segments are line segments from (xPoints[i - 1], yPoints[i - 1]) to (xPoints[i], yPoints[i]), for 1 ≤  *i* ≤  nPoints. The figure is automatically closed by drawing a line connecting the final point to the first point, if those points are different.

The area inside the polygon is defined using an even-odd fill rule, also known as the alternating rule.

**Parameters:**

xPoints - an array of x coordinates.

yPoints - an array of y coordinates.

nPoints - the total number of points.

* public void fillRect(int x,

int y,

int width,

int height)

Fill the specified rectangle. The left and right edges of the rectangle are at x and x + width - 1. The top and bottom edges are at y and y + height - 1. The resulting rectangle covers an area width pixels wide by height pixels tall. The rectangle is filled using the current color.

**Parameters:**

x - the *x*coordinate of the rectangle to be filled.

y - the *y*coordinate of the rectangle to be filled.

width - the width of the rectangle to be filled.

height - the height of the rectangle to be filled.

Coming week we are planning to start implementation of our project by dividing work among team members and designing in detail.