

**Assignment 2, Due Thurs 26 January 2006, 17.30pm. Worth 15% of CA.**

**Demonstrations of software execution, see below, must be completed by Thurs 26 January 2006, 17.30pm.**

This is a short assignment and can easily be completed in the laboratory hours of the next two Thursdays. I will be happy to provide assistance.

There will be two more assignments: one small one, worth 15% of CA, due Thurs 23 February 2006; and one larger one, worth 40% of CA, involving the 2D platform game, due 4th April 2006.

Late assignments without documented extenuating circumstances **will lose 10% of marks per day or part of day late. Those seven days late will receive zero marks.** If you have extenuating circumstances, you should contact me as soon as possible; in particular, be careful about wasting time on completing an assignment only to hand it in well after the deadline; whether you have valid reasons or not I cannot give any marks for work handed in after outline answers have been handed out or discussed in class.

Assignments may be typed or neatly handwritten; i.e. don't waste time typing if you don't want to; however the answers must be neat and tidy and easily followed (assignments may be inspected by external examiners). Please use only one side of the page. There is no need for ornate front sheets. Simply include the heading above (including my name), and your own name. A staple at the top left corner is the preferred binding – I can staple them if you wish. There is no need for plastic or other covering.

**Answers should be in your own words. For numerical answers, where appropriate, show your working; a numerical answer written down with no explanation of working is likely to get zero marks.**

0. (a) Create a directory in your `progs` directory called `assign2`;  
(b) Copy all of the contents of my Public Folder's `bscgp1\assign2\` into your `assign2`.
1. This is like item 4. of Practical 9. 100% of marks are for the demonstration.
  - (a) Compile `TileViewer1.java` and execute it.
  - (b) In order to fill out the frame, add the following filenames (they are repeated, but that will not affect the program):

```
fns.add("tile_A.png");    fns.add("tile_B.png");  
fns.add("tile_C.png");    fns.add("tile_D.png");  
fns.add("tile_E.png");    fns.add("tile_F.png");  
fns.add("tile_G.png");    fns.add("tile_H.png");  
fns.add("tile_I.png");    fns.add("star1.png");  
fns.add("star2.png");    fns.add("star3.png");  
fns.add("star4.png");    fns.add("heart1.png");  
fns.add("heart2.png");    fns.add("heart3.png");  
fns.add("music1.png");
```

Demonstrate working.

[10 marks]

- (c) Modify the program as described in Practical 9 to draw a black rectangle around each image. Demonstrate.

[10 marks]

2. 100% of marks are for the demonstration and code inspection.

Take `TileViewer1.java` and comment out the image display part. Insert the following code:

```

i= 0;
int red= 255, green= 255, blue= 255;
for(Image im : tiles){
    Color c = new Color(red, green, blue);
    g2.setColor(c);
    pt= pts.get(i); i++;
    /** decrement red, green, blue here red-= 20, ...
x1= (int)pt.getX();
y1= (int)pt.getY();
int w1= im.getWidth(null);
int h1= im.getHeight(null);
Rectangle rect = new Rectangle(x1, y1, 2, 2); /** replace 2,2
g2.fill(rect);
}

```

It is meant to draw a coloured (or grey) rectangle in place of each image. Start with white (255, 255, 255) and for each new rectangle decrement red, green, blue by 20 units.

[20 marks]

3. 100% of marks are for the demonstration and code inspection. This is similar to the *House* graphics programs we worked on before Christmas.

- (a) Compile `HouseViewer.java` and execute it.
- (b) Modify `HouseViewer.java` to (i) draw the House at (0,0) (already done), (ii) draw a rotated version of the house at 60° degree intervals from 60° to 300°. Use `AffineTransform`. Remove any unnecessary code. Use stippled lines for the rotated house:

```

Stroke stroke = new BasicStroke(1, BasicStroke.CAP_BUTT,
    BasicStroke.JOIN_BEVEL, 0, new float[] {3, 3}, 0);
g2.setStroke(stroke);

```

[20 marks]

4. 100% of marks are for the demonstration and code inspection.

- (a) Compile `HouseViewer1.java` and execute it; it is the same as `HouseViewer.java` but I want to avoid confusion with internal class names.
- (b) Modify `HouseViewer1.java` to (i) draw the House at (0,0) (already done), (ii) draw a translated version of the house at (100,0), (0,100), (100,100), (150,100), (100,150). Remove any unnecessary code. Use `AffineTransform` (`translate`). Use stippled lines for the translated house:

```

Stroke stroke = new BasicStroke(1, BasicStroke.CAP_BUTT,
    BasicStroke.JOIN_BEVEL, 0, new float[] {1, 1}, 0);
g2.setStroke(stroke);

```

[20 marks]

5. 100% of marks are for the demonstration and code inspection. This is based on the game Su Doku — but we are not building a game, only a prototype of a display for the game. I will provide a photocopy of a Su Doku game from a newspaper.

- (a) Compile `SudokuViewer.java` and execute it. You will see that it draws a  $9 \times 9$  grid.
- (b) Adjust the spacings of the grid lines to ensure that the grid cells are all the same size (obviously some of the Frame is taken up with borders and header). See `/**` in the code.

[5 marks]

- (c) In the Su Doku photocopy, you will notice that  $3 \times 3$  squares are identified by heavier lines; in fact, what we need is every third line drawn wider. Add this line to the `x` for loop; and add a similar line to the `y` for loop.

```

    if(i%3==0){
        l = new Line2D.Double(x+1, yLo, x+1, yHi);
        g2.draw(l);
    }

```

[10 marks]

- (d) Now we want to write numbers in the cells. After the `y` for loop, add code along the following lines to write numbers to each cell. 14 is the size of the Font; find a more appropriate size. Note: the numbers do not have to be a valid Su Doku solution, but they must be in the range 1 to 9.

```

    Font sb= new Font("SansSerif", Font.BOLD, 14);
    g2.setFont(sb);

    int num=1;
    for(int i= 0; i< n; i++){
        x= i*dx; // + ???;
        for(int j= 0; j< n; j++){
            y= j*dy; // + ???
            String label= ""+ num;
            num++;
            // if(num== ???) num=???
            g2.drawString(label, (float)x, (float)y);
        }
    }

```

[10 marks]

- (e) You should hand up a printed version of the final program for marking.

[20 marks]

6. In question 3., (`HouseViewer.java`), for the original House at the origin, the positions of the corners of the wall are  $(0, 0)$ ,  $(0, 80)$ ,  $(80, 0)$  and  $(80, 80)$ .

- (a) Using Octave (a matrix/vector calculation program available on 2213 machines), see below, determine the positions of each of these after a rotation by  $60^\circ$ . Copy and paste or otherwise show your answers with full working.

[30 marks]

- (b) Show the results of (a) in a simple diagram.

[10 marks]

- (c) Compute the inverse of the rotation matrix.

[5 marks]

- (d) Compute the product of the inverse of the rotation matrix with the rotation matrix itself. Comment on the result.

[5 marks]

## Octave example for rotating by 30 degrees.

```
octave:3> c = cos(30*pi/180.)
```

```
c = 0.86603
```

```
octave:4> s = sin(30*pi/180.)
```

```
s = 0.50000
```

```
octave:5> r30 = [c -s; s c]
```

```
r30 =
```

```
0.86603 -0.50000
```

```
0.50000 0.86603
```

```
octave:6> ir30 = inv(r30)
```

```
ir30 =
```

```
0.86603 0.50000
```

```
-0.50000 0.86603
```

```
octave:7> u =[5; 3]
```

```
u =
```

```
5
```

```
3
```

```
octave:8> v = r30*u
```

```
v =
```

```
2.8301
```

```
5.0981
```

```
octave:9> mat = r30*ir30
```

```
mat =
```

```
1.0000e+00 -6.2938e-17
```

```
4.0630e-17 1.0000e+00
```

Notice that if you want a *column* vector ( $2 \times 1$ ) you need the ';' in `u =[5; 3]`; if you used a '' it would be a *row* vector ( $1 \times 2$ ).