HSA_New Console Class

Constructors

Console ()	Constructor - Creates a Console window of 25 rows by 80 columns.
Console (int fontSize)	Constructor - Creates a Console window with the text size set to *fontSize*.
Console (int rows, int cols)	Constructor - Creates a Console window with width of *cols* columns and height of *rows* rows.
Console (int rows, int cols, int fontSize)	Constructor - Creates a Console window with width of *cols* columns, height of *rows* rows, and the text size set to *fontSize*.
Console (String title)	Constructor - Creates a Console window and sets the window title to *title*.
Console (int fontSize, String title)	Constructor - Creates a Console window with the text size set to *fontSize* and sets the window title to *title*.
Console (int rows, int cols, String title)	Constructor - Creates a Console window with width of *cols* columns, height of *rows* rows, and the window title set to *title*.
Console (int rows, int cols, int fontSize, String title)	Constructor - Creates a Console window with width of *cols* columns, height of *rows* rows, the text size set to *fontSize*, and the window title set to *title*.

Text Input and Output Methods

void **clear** ()	Clears the entire Console window and sets the cursor to the upper-left corner.
void **close**()	Closes the current console window.
int **getMaxColumns** ()	Returns the width of the Console window in columns.
int **getMaxRows** ()	Returns the height of the Console window in rows.
int **getColumn** ()	Returns the column number of the current cursor position.
int **getRow** ()	Returns the row number of the current cursor position.

Outputs the argument to the Console window beginning at the cursor position:

void **print** (byte b)	void **print** (short s)	void **print** (int i)	void **print** (long l)
void **print** (float f)	void **print** (double d)	void **print** (boolean b)	void **print** (char c)
void **print** (String s)			
void **println** (byte b)	void **println** (short s)	void **println** (int i)	void **println** (long l)
void **println** (float f)	void **println** (double d)	void **println** (boolean b)	void **println** (char c)
void **println** (String s)	void **println** ()		

byte **readByte** ()	Returns the byte value (-128 to 127) read from the keyboard.
short **readShort** ()	Returns the 2-byte integer (-32,768 to 32,767) read from the keyboard.
int **readInt** ()	Returns the 4-byte integer (-2,147,483,648 to 2,147,483,647) read from the keyboard.
long **readLong** ()	Returns the 8-byte integer read from the keyboard.
float **readFloat** ()	Returns the 4-byte float read from the keyboard.
double **readDouble** ()	Returns the 8-byte double read from the keyboard.
boolean **readBoolean** ()	Returns the boolean value (either **true** or **false**, case insensitive) read from the keyboard.
char **readChar** ()	Returns the character read from the keyboard.
String **readString** ()	Returns the String read from the keyboard.
String **readLine** ()	Returns the entire line of input read from the keyboard without the Return.

`void` **`setTextColor`** `(Color c)`	Sets the color for text output from print and println methods.
`void` **`setTextBackgroundColor`** `(Color c)`	Sets the background color for text output from print and println methods.
`void` **`showCursor`** `()`	Shows the text cursor, if invisible.
`void` **`hideCursor`** `()`	Hides the text cursor, if visible.
`void` **`setCursor`** `(int row, int col)`	Sets the cursor position to row *row* and column *col*.

Graphics Methods

`void` **`clearRect`** `(int x, int y, int width, int height)`
Clears the rectangle to the background color.

`void` **`copyArea`** `(int x, int y, int width, int height, int deltaX, int deltaY)`
Copies the rectangle defined by the upper-left corner (*x*, *y*) with width of *width* and height of *height* to a position moved by *deltaX* and *deltaY* pixels.

`void` **`draw3DRect`** `(int x, int y, int width, int height, boolean raised)`
Draws a 3-D rectangle. It appears raised if *raised* is true.

`void` **`drawArc`** `(int x, int y, int width, int height, int startAngle, int arcAngle)`
Draws an arc. The arc is inscribed in the rectangle defined by the upper- left corner (*x*, *y*) with width of *width* and height of *height*. It starts at *startAngle* degrees and goes counterclockwise for *arcAngle* degrees.

`void` **`drawImage`** `(Image i, int x, int y, ImageObserver observer)`
Draws as much of the specified image as is currently available at the point (*x*, *y*)

`void` **`drawImage`**`(Image img, int x, int y, int width, int height, ImageObserver observer)`
Draws the specified image at the point (*x*, *y*) with width of *width* and height of *height*.

`void` **`drawLine`** `(int x1, int y1, int x2, int y2)`
Draws a line from (*x1*, *y1*) to (*x2*, *y2*).

`void` **`drawMapleLeaf`** `(int x, int y, int width, int height)`
Draws a maple leaf. The maple leaf is inscribed in the rectangle defined by the upper-left corner (*x*, *y*) with width of *width* and height of *height*.

`void` **`drawOval`** `(int x, int y, int width, int height)`
Draws an ellipse. The ellipse is inscribed in the rectangle defined by the upper-left corner (*x*, *y*) with width of *width* and height of *height*.

`void` **`drawPolygon`** `(int[] xPoints, int[] yPoints, int numPoints)`
Draws a polygon. The *xPoints* and *yPoints* arrays define the coordinates of the array of vertices. *numPoints* specifies the number of vertices in the polygon.

`void` **`drawRect`** `(int x, int y, int width, int height)`
Draws a rectangle with upper-left corner at (*x*, *y*) with width of *width* and height of *height*.

`void` **`drawRoundRect`** `(int x, int y, int width, int height, int arcWidth, int arcHeight)`
Draws a rectangle with rounded corners with upper-left corner at (*x*, *y*) with width of *width* and height of *height*. *arcWidth* and *arcHeight* are the width and height of the ellipse used to draw the rounded corners.

`void` **`drawStar`** `(int x, int y, int width, int height)`
Draws a star. The star is inscribed in the rectangle defined by the upper-left corner (*x*, *y*) with width of *width* and height of *height*.

`void` **`drawString`** `(String str, int x, int y)`
Draws the string *str* at the starting point (*x*, *y*). The *y* coordinate is the base line of the text.

`void` **`fill3DRect`** `(int x, int y, int width, int height, boolean raised)`
Draws a filled 3-D rectangle. It appears raised if *raised* is true.

`void` **`fillArc`** `(int x, int y, int width, int height, int startAngle, int arcAngle)`
Draws a filled arc. The arc is inscribed in the rectangle defined by the upper-left corner (*x*, *y*) with width of *width* and height of *height*. It starts at *startAngle* degrees and goes counterclockwise for *arcAngle* degrees.

`void` **`fillMapleLeaf`** `(int x, int y, int width, int height)`
Draws a filled maple leaf. The maple leaf is inscribed in the rectangle defined by the upper-left corner (*x*, *y*) with width of *width* and height of *height*.

`void` **`fillOval`** `(int x, int y, int width, int height)`
Draws a filled ellipse. The ellipse is inscribed in the rectangle defined by the upper-left corner (*x*, *y*) with width of *width* and height of *height*.

`void` **`fillPolygon`** `(int[] xPoints, int[] yPoints, int numPoints)`
Draws a filled polygon. The *xPoints* and *yPoints* arrays define the coordinates of the array of vertices. *numPoints* specifies the number of vertices in the polygon.

`void` **`fillRect`** `(int x, int y, int width, int height)`
Draws a filled rectangle with upper-left corner at (*x*, *y*) with width of *width* and height of *height*.

`void` **`fillRoundRect`** `(int x, int y, int width, int height, int arcWidth, int arcHeight)`
Draws a filled rectangle with rounded corners with upper-left corner at (*x*, *y*) with width of *width* and height of *height*. *arcWidth* and *arcHeight* are the width and height of the ellipse used to draw the rounded corners.

`void` **`fillStar`** `(int x, int y, int width, int height)`
Draws a filled star. The star is inscribed in the rectangle defined by the upper-left corner (*x*, *y*) with width of *width* and height of *height*.

`int` **`getWidth`** `()`	Returns the width of the Console window in pixels.
`int` **`getHeight`** `()`	Returns the height of the Console window in pixels.
`void` **`setColor`** `(Color c)`	Sets the color of the graphics context. The color is used for any draw methods.
`void` **`setFont`** `(Font f)`	Sets the font of the graphics context. The font is used with the drawString method.
`void` **`setPaintMode`** `()`	Sets the graphics context into paint mode. All drawing in the graphics context draws over the background.

`void` **`setXORMode`** `(Color c)`
Sets the graphics context into XOR mode. All drawing in the graphics context is XOR'd with the background. The color specified by c is a special color so that any drawing done on a background of color c will not be changed.

Example

```
import java.awt.*;
import hsa_new.Console;

public class TestConsole
{
    public static void main (String[] args)
    {
        Console c = new Console ();
        c.print ("Enter the radius of the circle: ");
        int radius = c.readInt ();
        c.setColor (Color.green);
        c.fillOval (c.getWidth () / 2 - radius, c.getHeight () / 2 - radius,
            radius * 2, radius * 2);
    } // main method
} /* TestConsole class */
```

HSA_New Message Class

Constructors and Methods

Message (String message)
: Constructor - Displays *message* in a untitled window centered on the screen. When the **OK** button is pressed, execution returns from the call to the constructor.

Message (String message, String title)
: Constructor - Displays *message* in a window titled *title* centered on the screen. When the **OK** button is pressed, execution returns from the call to the constructor.

Message (String message, Frame frame)
: Constructor - Displays *message* in a untitled window centered on Frame *frame*. When the **OK** button is pressed, execution returns from the call to the constructor.

Message (String message, String title, Frame frame)
: Constructor - Displays *message* in a window titled *title* centered on Frame *frame*. When the **OK** button is pressed, execution returns from the call to the constructor.

static void **beep** ()
: Causes the computer to beep. Useful for catching the user's attention.

Example

```
import hsa_new.Message;

    // Display four messages, one after another.
    public class TestMessage
    {
        public static void main (String[] args)
        {
            Message.beep();
            new Message ("This is the first message", "First message");
            new Message ("This is the second message", "Second message");
            new Message ("This is the third message", "Third message");
            new Message ("This is the last message", "Last message");
        } // main method
    } /* TestMessage class */
```