건드려 보장 XTextField

안녕하세요 핑크레드 입니다.^^; 늦었지만 새해 복 많이 받으시고 올해에는 부자 되십시오..연봉 팍팍 올라가십시오..ㅋㅋㅋ 컨텐츠 대박 되시고..기타 등등..

어느 날 갑자기 지하철을 탔는데..제가 문 앞에 바짝 붙어서서 내릴 준비를 하고 있었습니다. 그것도 한 정거장 뒤에서 ㅡㅡ; 세삼 내가 왜 이렇게 조바심 내면서 살고 있는지..한걸음 뒤로 물러섰습니다. 그리고 얼마 후에 또 바보같이 하늘을 쳐다보았습니다. 아파트 위로 높은 건물들이 보이더군요..와 세상은 높구나 라는 것을 느꼈습니다. 내가 왜 매일 프로그램을 짜고 있는지 또 그런 생각을 했습니다. 요즘 긴장이 점 풀렸습니다. 우물안의 개구리라고 할 수 있을지..한번쯤은 이런 생각도 해보는 것도 좋을듯합니다.

오늘은 프로그램의 세계로 떠나 보도록 해봅시다.. 다들 많은 프로그램을 해보셨지만 프로그램이라 함은 하나의 예술이라고 생각합니다. 그럼 예술은 어떻게 할까요. ㅋㅋㅋ 예술의 시작은 바로 새로운 이미테이션이라고 할 수 있죠. 전체적인 모습을 상상하고 스케치를 하고, 밑그림을 그리고, 예술로 승화시키는 거죠..무슨 쓸 때 없는 말이냐구요.. 제가 한때는 예술을 몰랐습니다. 그런데 이제서야 예술의 의미를 조금 깨닫고 있습니다. 프로그래머 초짜 시절 전 작은 것에서 큰 것을 만들어 냈습니다. 이렇게 하다 보니 짜면짤수록 스팩이 커지고 필요한것들도 자꾸 생기고 그렇다군요..그러다가 논문땜시 전체로부터 시작해서 조그마한 부분으로 프로그램을 했었죠.. 그것이 시작이었습니다. 모든 이치는 똑같았죠 전 이미 프로그램의 모든 모습을 보고 있었던 것입니다. 상상속의 예술을요..
이것을 대학교 3 학년 말에 가서 느꼈죠..ㅡㅡ; 바부 같이..
다들 아는 것이지만 보통 모바일 프로그램은 많아야 2 명, 보통 혼자서 거의 프로그램을 합니다. 또한 기획자도 따로 있기 때문에 프로그래머는 그냥 앉아서 손가락 노가다만 하게 됩니다. 가끔씩 우리 예술을 해보조..ㅋㅋㅋ 쓸 때 없는 말이었습니다. 하늘이란 것은 그다지 높지는 않답니다.

얼마전에 별루 안됐죠. .XCE 에서 말도 많고 탈도 많았던 XTextField 를 공개했습니다. 아마 오래전부터 공개한다고 했는데 이제 하네요..그래도 지금이나마 공개해서 다행이라고 해야하나..걱정이라고 해야하나.. 여튼 소스가 공개됨으로서 많은 부분을 변경 가능하게 되었습니다. 멀티라인은 물론 엔터처리 글자색 바꾸기, 커서 모양 및 색 바꾸기..등등. 프로그래머로서는 한마디로 기쁘죠..재창조의 기쁨이죠. 일단 XTextField 를 나가지 전에 또 우리의 팁들을 대령해야 겠죠. 오늘은 아주 충격먹을 만한 팁을 알려드리겠습니다. 웬만하면 이 팁은 절대로 공개 안할려고 했는데 이제 때가 때이니만큼 제 강좌를 널리 이롭게 하기 위해서.. ㅋㅋㅋ 오늘 이 강좌 보시는 분들 운 좋은 겁니다. 일단 한번 봐봐 뽕갑니다. 시작은 거

창하지만 알만한 사람은 다아는 팁입니다. (ㅡㅡ)ㄱ

컨텐츠를 제작함에 있어서 아래 3 가지를 줄이고 어떻게 해볼려고 난리를 칩니다.

1. 컨텐츠 용량
2. 메모리
3. 속도

이 3 박자가 딱딱 맞아 떨어져야 우리는 희열을 느낍니다. 근데 이게 어디 쉬운일입니까?
이 3 가지를 그나마 효율적으로 다룰 수 있는 부분을 발견하여 모두들에게 이롭게 하려는 것입니다. 그 비밀의 팁을 알려드리도록 하겠습니다.

우리가 컨텐츠를 하는데 있어서 가장 용량을 많이 차지하는 것이 무엇일까요? 그건 바로 이미지 입니다. 이미지만큼 용량을 많이 차지하고 메모리를 잡아 먹는 것은 없지요. 그게 바로 핵심 포인트 입니다. 바로 이 이미지를 삶아 먹으면 우리의 고민이 해결되는 것이지요.

방법이 무어냐..이 방법은 이미지를 로딩하는데 있습니다. 일반적으로 이미지를 불러오는 것이 느린데 이유는 다음과 같습니다.

1. 파일을 엑섹스 하기 때문에
2. 개수가 여러 개 이기 때문에

이와 같은 이유가 이미지를 로딩하는데 느린 이유입니다. 원래 단말기는 파일을 엑섹스 하는 속도가 상당히 느립니다. 거기다가 개수도 여러 개이니 당근 느릴수 밖에요. 그럼 어떻게 하는냐 하나의 이미지에 여러 이미지를 합치는 방법이 있겠지요. 하지만 이미지의 크기가 전부 다르기 때문에 하나로 합치는데 문제가 있고 또한 그 이미지가 어디 영역의 있는지를 체크해야 합니다. 무지 귀찮습니다. 이것을 해결할 수 있는 메소드가 바로 이것입니다.

짜잔…

createImage(byte[] imageData, int imageOffset, int imageLength)

바로 Offset 을 계산해서 이미지를 로딩하는 것입니다. 이해가 안가신다구요..예를 들어 이미지가 10 장이 있는데 각각의 이미지의 크기가 다릅니다. 이 이미지들을 10 개의 파일을 하나의 파일로 만드는 것입니다. 어떻게 만드느냐..뒤로 이어서 붙이는 겁니다. 이렇게 하면 이미지의 크기와 상관없이 합칠 수 있습니다.
예시)

1. 첫번째.lbm : 500byte
2. 두번째.lbm : 200byte
3. 세번째.lbm : 234byte

4. 네번째.lbm : 601byte

….

10. 열번째.lbm 9332byte

라고 한다면 해당 데이터를 아래와 같이 하나의 파일로 만드는 겁니다.

500	200	234	601	…	…		9332

그리고 나서 해당 데이터를 읽습니다.

```
InputStream is = getClass().getResourceAsStream("/img/loading/MixImage_1.dat");
byte[] imageData = new byte[is.available()];
is.read(imageData);
is.close();
```

그리고 나서 다음 해당 API 를 사용합니다.

```
imgFirst = Image.createImage(imageData, 0, 500);
imgSecond = Image.createImage(imageData, 500, 200);
imgThird = Image.createImage(imageData, 700, 234);
…..
imgTenth = Image.createImage(imageData, xxx, 9332);
imageData = null;
```

이런식으로 이미지를 사용하게 되면 얻는 이득은 설명한 바와 같습니다. 부득이 설명하자면 파일이 하나이니 파일 엑섹스 시간이 줄고, 큰 덩어리의 파일이기 때문에 메모리 단편화를 줄일 수 있고, jar 로 압축 시에도 기존보다 용량이 줄어듭니다. 그러므로 이 방법을 권해 드립니다. 여기서 걸리는 문제는 이렇게 대용량이다 보니 이미지를 처음에 로딩하는 컨텐츠들이 사용시에 그 효과가 극대화댑니다. 중간중간 한다면 아마 단편화로 고생합니다. 그러므로 꼭 처음에 이미지를 거의 로딩하는 컨텐츠에 대해서 사용하시는 것이 좋을 듯 합니다.
주의 사항으로는 하나의 이미지 데이터가 약 50kb 정도 단위로 짤라서 작성해 주시면 좋을 듯 합니다. 한동안 작업을 안해서 잘 모르겠는데요 제가 할 때는 50kb 정도로 잘라서 하니 적당히 메모리 풀 안나고 잘되더라구요. 그것은 컨텐츠마다 다르니 적당량을 잡으시는 것이 좋을 듯 합니다. 그럼 여기서 생기는 문제는 저것을 일일이 계속 한다면 얼마나 귀찮을까 입니다. 프로그램을 만드셔서 하십시오. 이 부분에서는 XCE 쪽에 문의 결과 좋은 방법이다라고 하더군요. 또한 XCE 에서는 이미 해당 사항을 알고 프로그램까지 가지고 있다고 합니다. 그래서 해당 프로그램 및 사용방법을 따로 올리신다고 공지해 주었습니다. 그러므로 여러분들은 XCE 에서 올려주시면 그냥 사용방법만 알고 쓰시면 될 것 같습니다.

좋은 팁이 되었으면 하는 바람입니다. 전 처음에 이거 쓰고 뽕갔습니다.
마약과 같았죠 ㅡㅡ; 계속 쓰고 있습니다..ㅋㅋㅋ
이제는 본론으로 들어가서 XTextField 에 대해서 알아 보도록 하겠습니다. 일단 기존 것을 어떻게 요리하냐가 관건입니다. 무엇을 추가할지 무엇을 없앨지..기타등등

일단 무엇을 위한 XTextField 냐 단순한 입력창이다 라고 생각하면 이래저래 필요 없는 부분 전부 없애야하고, 특별한 기능을 위한 것이다라고 한다면 전부 구현해 주어야 합니다.

언제나 기본 스펙을 가지고 요리를 하시면 되겠습니다.
다음 소스는 기본 XTextField 를 가지고 커서 모양, 깜빡임, 엔드라인 처리 등등

이모티콘에 필요한 기능들을 구현한 것입니다. 한번 보시고 필요한 기능들을 추가하시거나 혹은 수정하시거나 하시면 되겠습니다. 특별히 XTC-10 번 문서도 참고 하시면 더할 나위 없습니다. 그리고 제가 중간중간 실수한 부분이 있더라도 이해와 관용을.. ㅡㅡ;

```
import javax.microedition.lcdui.*;
import com.xce.lcdui.*;

public class PinkTextField
{
	char[] text;                                // Text 문장
	private int textCount;                      // 현재 문자 개수

	private int[] lineIndex = new int[20];      // 라인수(최대 20)
	private int lineCount = 1;                  // 초기 라인 개수

	private int x;                  // Text Box X 좌표 위치
	private int y;                  // Text Box Y 좌표 위치
	private int width;              // TextBox 폭
	private int height;             // TextBox 높이

	private int constraints;
	private int maxSize;            // 최대 문자 개수
	private int caretPos;           // 커서 위치
	private int curX, curY;         // 커서의 X, Y 좌표

	// Repaint 관련
	private int startLine;      // 시작라인
	private int maxLine;        // 최대 라인
	private int caretLine;      // 현재 커서 라인

	private int textColor;      // 텍스트 색상
	private int cursorColor;    // 커서 색상
	private int cursorTextColor;    // 커서안 텍스트 색상

	//scroller
```

```
        private int visHeight;          // 스크롤바 높이
        private int scrollX = 0;        // 스크롤바 X 좌표
        private int scrollY = 0;        // 스크롤바 Y 좌표

        private boolean focus;          // 포커스 유무

        PinkCanvas pinkCanvas;          // 캔버스

        private static final TextComponentHandler imListener =
TextComponentHandler.getTextComponentHandler();
        private final InputMethodImpl imi = new InputMethodImpl(this);

        // 아래 부분은 고칠 것이 없습니다. ㅡㅡ;
        class InputMethodImpl implements TextComponent
        {
            PinkTextField pinkTextField;

                InputMethodImpl(PinkTextField _pinkTextField)
                {
                        pinkTextField = _pinkTextField;
                }

                public int getCaretPosition()
                {
                        return pinkTextField.getCaretPosition();
                }

                public int getConstraints()
                {
                        return pinkTextField.getConstraints();
                }

                public int getMaxSize()
                {
                        return pinkTextField.getMaxSize();
                }

                public int size()
                {
                        return pinkTextField.size();
                }

                public void insert(char data)
                {
                        pinkTextField.insertChar(data);
                }

                public void delete()
                {
                        pinkTextField.deleteChar();
                }

                public void clear()
                {
                        pinkTextField.setString("");
                }
```

```
		public void replace(char data)
		{
			pinkTextField.replaceChar(data, caretPos - 1);
		}

		 // 커서가 움직임 적용
		public void moveCursor(int keyCode)
		{
			if (textCount >= 0 )
			{
				switch (keyCode)
				{
					case Canvas.KEY_LEFT:		//	Key
Left
						moveLeftCaret();
						break;
					case Canvas.KEY_RIGHT:		//
KEY_RIGHT
						moveRightCaret();
						break;
					case Canvas.KEY_UP:		// Key Up
						moveUpCaret();
						break;
					case Canvas.KEY_DOWN:
	//KEY_DOWN
						moveDownCaret();
						break;
				}
			}
			 // 커서의 위치를 업데이트 하고
			pinkTextField.updateCaret();
			// 입력된 내용을 삭제합니다.
			imListener.clear();
		}

		// 현재 커서의 위치를 설정합니다.
		public void setCaretPosition(int index)
		{
			if(index >= 0 && index <= textCount)
			{
				caretPos = index;

				for (int i = 0; i < lineCount; i++)
				{
					if (caretPos <= lineIndex[i])
					{
						caretLine = i;
						break;
					}
				}
				updateCaretLine();
			}
		}

		public void setCaretVisible(boolean flag) {}
```

```
		public void repaint()
		{
		}

		public void repaintIM()
		{
		}
	}

	//한번 쓰고 버리면 모르지만 재활용을 위해서 대비하는 것입니다.
	public PinkTextField(PinkCanvas _pinkCanvas)
	{
		pinkCanvas = _pinkCanvas;
	}

	public PinkTextField(PinkCanvas _pinkCanvas, String text2, int maxSize, int constraints)
	{
		pinkCanvas = _pinkCanvas;

		text = text2.toCharArray();
		caretPos = textCount = text.length;

		this.setMaxSize(maxSize);
		this.setConstraints(constraints);
	}

/**************************************************************************
	Key Input Event
 **************************************************************************/
	public void keyPressed(int keyCode)
	{
		imListener.keyPressed(keyCode);
	}

	public void keyReleased(int keyCode)
	{
		imListener.keyReleased(keyCode);
	}

	public void keyRepeated(int keyCode)
	{
		imListener.keyRepeated(keyCode);
	}


/**************************************************************************
	Painting
 **************************************************************************/

	public void paint(Graphics g)
	{
		if(x == 0 && y == 0 && width == 0 &&   height == 0)
			return;

		paintChars(g);		// 글자 그리기
```

```
                updateCaret();
                paintCaret(g, false, 0);          // 커서 그리기
        }

        // 커서 그리기 입니다.
         // 가운데 인자는 커서를 깜빡이게 하는 것입니다.
         // 깜빡임을 위한 템포 입니다. 홀수와 짝수를 두어서 깜빡이게 했습니다.
         // 커서의 경우에 따라서 모양을 달리했습니다.
         // 1. 글자 위에 있을 경우, 2. 스페이스 일경우, 3. 문장끝에 있을경우
        public void paintCaret(Graphics g, boolean isFlash, int tempo)
        {
                if(focus)
                {
                        int caretWidth = 0;
                        if(lineIndex[caretLine] == caretPos || textCount == 0)
                        {
                                caretWidth = Toolkit.DEFAULT_FONT.charWidth(' ');

                                if(isFlash && (tempo % 2 == 0))
                                {
                                        g.setColor(0xffffff);
                                        g.fillRoundRect(curX, curY +
                                   (Toolkit.FONT_HEIGHT - 12) / 2, caretWidth, 12, 2, 2);

                                        return;
                                }

                                g.setColor(cursorColor);
                                g.fillRoundRect(curX, curY + (Toolkit.FONT_HEIGHT -
12) / 2, caretWidth, 12, 2, 2);
                        } else if(caretPos + 1 == maxSize)
                        {
                                caretPos++;
                                updateCaret();

                                caretWidth = Toolkit.DEFAULT_FONT.charWidth(' ');

                                if(isFlash && (tempo % 2 == 0))
                                {
                                        g.setColor(0xffffff);
                                        g.fillRoundRect(curX,          curY          +
(Toolkit.FONT_HEIGHT - 12) / 2, caretWidth, 12, 2, 2);

                                        return;
                                }

                                g.setColor(cursorColor);
                                g.fillRoundRect(curX, curY + (Toolkit.FONT_HEIGHT -
12) / 2, caretWidth, 12, 2, 2);
                        } else
                        {
                                if(isFlash && (tempo % 2 == 0))
                                {
                                        g.setColor(0xffffff);
                                        caretWidth     =     (text[caretPos]     ==     '
')?Toolkit.DEFAULT_FONT.charWidth(' '):Toolkit.DEFAULT_FONT.charWidth(text[caretPos]);
```

```
                                        g.fillRoundRect(curX, curY, caretWidth, Toolkit.FONT_HEIGHT, 2, 2);

                                        g.setColor(textColor);
                                        if(text[caretPos] != ' ')
                                                g.drawChar(text[caretPos], curX, curY, g.TOP|g.LEFT);

                                        return;
                                }

                                g.setColor(cursorColor);
                                caretWidth = (text[caretPos] == ' ')?Toolkit.DEFAULT_FONT.charWidth(' '):Toolkit.DEFAULT_FONT.charWidth(text[caretPos]);
                                g.fillRoundRect(curX, curY, caretWidth, Toolkit.FONT_HEIGHT, 2, 2);

                                g.setColor(cursorTextColor);
                                if(text[caretPos] != ' ')
                                        g.drawChar(text[caretPos], curX, curY, g.TOP|g.LEFT);
                        }
                }
        }

        // 스크롤바를 따로 구현하였습니다.
        // 스크롤바 모양은 언제든지 바꾸시면 됩니다.
        public void paintScrollBar(Graphics g, int _x)
        {
                if(caretLine < (height / Toolkit.FONT_HEIGHT))
                        return;

                g.setColor(171, 181, 127);
                g.drawRect(_x, y + 15, 5, height - 26);

                g.setColor(230, 235, 209);
                g.drawRect(_x + 1, y + 15 + 1, 3, height - 28);

                g.setColor(203, 211, 171);
                g.drawRect(_x + 2, y + 15 + 2, 1, height - 30);

                int _y = y + 16 + (height - 32) * startLine / (caretLine - (height / Toolkit.FONT_HEIGHT) + 1);

                g.setColor(152, 163, 14);
                g.drawLine(_x + 1, _y, _x + 1, _y + 4);
                g.drawLine(_x + 1, _y, _x + 4, _y);
                g.setColor(161, 194, 0);
                g.fillRect(_x + 2, _y + 1, 2, 3);
                g.setColor(117, 126, 14);
                g.drawLine(_x + 4, _y + 1, _x + 4, _y + 3);
                g.drawLine(_x + 1, _y + 4, _x + 4, _y + 4);
        }

        // 그냥 넘어 갑니다. ㅡㅡ;
        public void setTextColor(int _textColor)
```

```
{
        textColor = _textColor;
}

public void setCursorColor(int _cursorColor)
{
        cursorColor = _cursorColor;
}

public void setCursorTextColor(int _cursorTextColor)
{
        cursorTextColor = _cursorTextColor;
}

 // 문자 그리기
private void paintChars(Graphics g)
{
        if (textCount == 0 )
                return;

        int cx = x;
        int cy = y;
        int drawnLineCount = 0;

        if( focus )
                g.setColor(textColor);
        else
                g.setColor(0x000000);

       // 한줄에 표현할수 있는 글자의 폭을 계산한 후에 화면에 보여줍니다.
        for (int i = startLine; i < lineCount && drawnLineCount < maxLine; ++i)
        {
                cx = x;

                if (i == 0)
                {       // 글자 길이를 계산합니다.
                        for(int j = 0, k = 0; j < lineIndex[i]; j++)
                        {
                                g.drawChar(text[j], cx, cy, Graphics.LEFT |
                                  Graphics.TOP);
                                cx +=
                                  Toolkit.DEFAULT_FONT.charWidth(text[j]);
                        }
                }else
                {
                        for(int j = lineIndex[i - 1], k = 0; j < lineIndex[i]; j++)
                        {
                                g.drawChar(text[j], cx, cy, Graphics.LEFT |
                                   Graphics.TOP);
                                cx +=
                                     Toolkit.DEFAULT_FONT.charWidth(text[j]);
                        }
                }

                cy += Toolkit.FONT_HEIGHT;
                drawnLineCount++;
```

```
                }
        }


/*****************************************************************************
        ScrollBar
 *****************************************************************************/
        // 스크롤 바의 위치를 설정합니다.
        // 일반적으로 전체 줄수 / 현재 커서 위치
        private void updateScrollBar()
        {
                int height2 = height;
                int viewPortLine = height2/Toolkit.FONT_HEIGHT;

                if( lineCount > viewPortLine )
                {
                        scrollY = height2 * startLine / lineCount;
                        if( startLine+viewPortLine >= lineCount )
                                visHeight = height2 - scrollY;
                        else
                                visHeight = height2 * viewPortLine / lineCount;
                } else
                {
                        scrollY = 0;
                        visHeight = height2;
                }
        }

        public void setScrollBar(boolean addScrollBar)
        {
                if(addScrollBar)
                        scrollX = x + width - 1;
                else
                        scrollX = 0;

                updateScrollBar();
        }


/*****************************************************************************
        내부적인 데이터 관리
 *****************************************************************************/

    public void insert(char[] data, int offset, int length, int position)
    {
                if (data == null)
                        throw new NullPointerException();

                if (offset < 0 || length < 0 || (offset + length) > data.length )
                        throw new ArrayIndexOutOfBoundsException();

                if (textCount + length > maxSize)
                        throw new IllegalArgumentException();

                 // char 카피시에는 다음과 같이 하는 것이 빠릅니다.
                System.arraycopy(text, position, text, position + length, textCount - position);
```

```
                System.arraycopy(data, 0, text, position, length);

                textCount += length;

                if (caretPos >= position)
                        caretPos += length;

                imListener.clear();
                this.doLayout(0);
        }

        public void insert(char data)
        {
                insertChar(data);
        }

        public void insert(String src, int position)
        {
                if (src == null)
                        throw new NullPointerException();

                this.insert(src.toCharArray(), 0, src.length(), position);
        }

        // 문자를 set 하는 부분이죠.
        public void setChars(char[] data, int offset, int length)
        {
                if (data == null)
                        textCount = caretPos = 0;
                else
                {
                        // 에러를 검증하는 부분이 많은데…그냥 해도 될지도.
                        if (offset < 0 || length < 0 || (offset + length) > data.length )
                                throw new ArrayIndexOutOfBoundsException();

                        if (length > maxSize)
                                throw new IllegalArgumentException();

                        System.arraycopy(data, offset, text, 0, length);
                        textCount = length;

                        caretPos = (this.getBytes()== maxSize)?textCount - 1:textCount;
                }

                imListener.clear();
                this.doLayout(0);
                startLine = 0;
        }

        public void setString(String data)
        {
                if (data == null || data.getBytes().length > maxSize)
                        throw new IllegalArgumentException();

                this.setChars(data.toCharArray(), 0, data.length());
        }
```

```
	// 문자를 삭제 합니다.
	public void delete(int offset, int length)
	{
		if (offset < 0 || length < 0 )
			throw new StringIndexOutOfBoundsException();

		if( offset + length > textCount )
			length = textCount - offset;

		System.arraycopy(text, offset + length, text, offset, textCount - offset -
length);
		textCount -= length;

		caretPos -= length;
		imListener.clear();
		this.doLayout(0);
	}

	public String getString()
	{
		return (textCount == 0)?"":String.valueOf(text, 0, textCount);
	}

/**************************************************************************
	Layout 관련
 **************************************************************************/

	private void doLayout(int fromLine)
	{
		if (width == 0)
			return;

		int currLineLen = 0;
		int currLineByte = 0;
		int currLineCount = 0;
		int startPos = fromLine == 0 ? 0 : lineIndex[fromLine - 1];
		int indexLine = fromLine;
		lineCount = fromLine + 1;

		boolean changeCaretLine = false;
		boolean enter = false;
		// 현재 커서 위치가 시작 위치보다 큰경우
		if (caretPos > startPos)
			changeCaretLine = true;

		while (startPos < textCount)
		{
			currLineLen = 0;
			currLineByte = 0;
			currLineCount = 0;
			enter = false;

			 // “ \n” 일경우를 개행문자로씀
			for (int i =startPos; i < textCount; i++)
			{
				if(text[i] == '\n')
```

```
                                        {
                                                enter = true;
                                                break;
                                        }

                                        currLineCount++;

                                        // 문자의 가로 크기로 한줄의 Count 체크
                                        currLineLen +=
                                                Toolkit.DEFAULT_FONT.charWidth(text[i]);

                                        if (currLineByte > 16 || currLineLen > width)
                                        {
                                                currLineCount = i - startPos;
                                                break;
                                        }
                                }

                                startPos += (currLineCount + ((enter)?1:0));

                                if (changeCaretLine && startPos >= caretPos)
                                {
                                        caretLine = indexLine;
                                        changeCaretLine = false;
                                }

                                lineIndex[indexLine++] = startPos;

                                if (indexLine > lineCount)
                                        lineCount++;
                        }
                        updateCaretLine();
                }

                private void setLineStart(int index, int start)
                {
                        if (index >= lineIndex.length)
                        {
                                int[] lineIndex2 = new int[lineIndex.length + 4];
                                System.arraycopy(lineIndex, 0, lineIndex2, 0, lineIndex.length);
                                lineIndex = lineIndex2;
                        }
                        lineIndex[index] = start;
                }

                private int getLineStart(int index)
                {
                        return lineIndex[index];
                }

                public void setBounds(int x, int y, int _width, int _height)
                {
                        this.x = x;
                        this.y = y;
                        this.width = _width;
                        this.height = _height;
```

```
                maxLine = (int)(height / Toolkit.FONT_HEIGHT);

                doLayout(0);
        }

 /*************************************************************************
        TextComponentHandler 에서 호출되는 메소드
 *************************************************************************/

        private void insertChar(char data)
        {
                if(this.getBytes() + String.valueOf(data).getBytes().length > maxSize)
                {
                        imListener.clear();
                        return;
                }

                if (caretPos <= textCount && textCount < maxSize)
                {
                        System.arraycopy(text, caretPos, text, caretPos + 1, textCount -
caretPos);
                        text[caretPos++] = data;
                        textCount++;
                        this.doLayout(caretLine);
                }
        }

        private void deleteChar()
        {
                if (caretPos != 0)
                {
                        System.arraycopy(text, caretPos, text, caretPos - 1, textCount -
caretPos);
                        textCount--;
                        caretPos--;
                        this.doLayout(caretLine == 0 ? 0 : caretLine - 1);
        // 한줄 위로 올라갔을 경우
                }
        }

        private void replaceChar(char data, int pos)
        {
                if(this.getBytes() > maxSize)
                        return;

                text[pos] = data;
                this.doLayout(caretLine);
        }

/*************************************************************************
        Update
 *************************************************************************/

        private void updateCaretLine()
        {
```

```
                if (caretLine >= lineCount)     // setString()해서 caretLine 이 줄었을때
                        caretLine = lineCount - 1;

                if (startLine + maxLine <= caretLine)
                {
                        startLine = caretLine - (maxLine - 1);
                }
                else if (startLine > caretLine)
                {
                        startLine = caretLine;
                }

                if(scrollX > 0)
                        updateScrollBar();
        }

        private void updateCaret()
        {
                int index = caretLine == 0 ? 0 : lineIndex[caretLine - 1];
                int lineWidth = 0;

                for(int i = index; i < caretPos; i++)
                {
                        if((byte)text[index] == (byte)'\n')
                                break;
                        else
                                lineWidth += Toolkit.DEFAULT_FONT.charWidth(text[i]);

                }

                curY = y + Toolkit.FONT_HEIGHT*(caretLine-startLine);
                curX = x + lineWidth;
        }

/*************************************************************************
        Text Size
 *************************************************************************/

        public void setMaxSize(int maxSize)
        {
                this.maxSize = maxSize;
                caretPos = textCount = 0;
                text = null;
                text = new char[maxSize];
                imListener.clear();
        }

        public int getMaxSize() {       return maxSize;}
        public int size() { return textCount;}

/*************************************************************************
        Caret 관련
 *************************************************************************/

        // 좌우상하로 움직일경우 개행문자로 인하여 길이 위치 등을 고려하여 움직임
        private void moveLeftCaret()
```

```
	{
		if (caretPos > 0)
		{
			caretPos--;
			if (caretLine != 0 && lineIndex[caretLine - 1] > caretPos)
			{		//위치가 라인의 첫번째 보다 작으면 라인을 올림
				caretLine--;

				 // 개행 문자가 나오면 넘김
				if(text[caretPos] == '\n')
					caretPos--;

				updateCaretLine();
			}
		}
	}
	// 오른쪽으로 커서를 옮김
	private void moveRightCaret()
	{
		if (caretPos < textCount)
		{
			caretPos++;
			// 개행 문자일 경우 오른쪽으로 옮김 그리고 한라인 내림
			if(text[caretPos] == '\n')
			{
				caretLine++;
				caretPos++;
				updateCaretLine();
			// 개행 문자가 아니다러도 폭이 차면 아래로 내림
			} else if (lineIndex[caretLine] <= caretPos)
			{
				caretLine++;
				updateCaretLine();
			}
		} else
		{		// 만일 문자가 없으면 빈 공간 넣기
			if(textCount < maxSize)
			{
				insertChar(' ');
			}
		}
	}
	// 커서를 위로 올릴경우
	private void moveUpCaret()
	{
		if (caretLine <= 0)
			return;
	// 커서를 위올릴경우 폭을 계산해서 위로 올림
		int	toCaretWidth	=	Toolkit.DEFAULT_FONT.charsWidth(text,
lineIndex[caretLine - 1],
                                                                caretPos -
lineIndex[caretLine - 1]);
		caretLine--;

		int caretLineStart = caretLine == 0 ? 0 : lineIndex[caretLine - 1];
		int toCaretWidth2 = 0;
```

```
                int i = caretLineStart;
                // 개행 문자가 나올경우 한칸뒤로..
                for (; toCaretWidth2 < toCaretWidth && i < lineIndex[caretLine]; i++)
                {
                        if(text[i + 1] == '\n')
                                break;

                        toCaretWidth2 += Toolkit.DEFAULT_FONT.charWidth(text[i]);

                }

                caretPos = i;
                updateCaretLine();
        }
        // 커서를 아래로 내릴경우
        private void moveDownCaret()
        {
                if (caretLine >= lineCount - 1)
                        return;
                // 역시 폭을 계싼함
                int caretLineStart = caretLine == 0 ? 0 : lineIndex[caretLine - 1];
                int toCaretWidth = Toolkit.DEFAULT_FONT.charsWidth(text, caretLineStart,
                                                                        caretPos -
caretLineStart);
                caretLine++;

                caretLineStart = lineIndex[caretLine - 1];
                int toCaretWidth2 = 0;
                int i = caretLineStart;
                 // 개행문자일 경우 역시 고려함
                for (; toCaretWidth2 < toCaretWidth && i < lineIndex[caretLine]; i++)
                {
                        if(text[i + 1] == '\n')
                                break;

                        toCaretWidth2 += Toolkit.DEFAULT_FONT.charWidth(text[i]);
                }
                caretPos = (i > lineIndex[caretLine])? i - 1 : i;
                updateCaretLine();
        }

       // 전체적인 문장의 시작을 한칸 올림
        public void moveDown()
        {
                if(startLine < caretLine - (height / EmoticonMain.FONT_HEIGHT) + 1)
                        startLine++;
        }
       // 전체적인 문장의 시작을 한칸 내림
        public void moveUp()
        {
                if(startLine > 0)
                        startLine--;
        }

/*
        public boolean isFirstLine()
```

```
		{
			if( caretLine <= 0 )
			{
				return true;
			}
			return false;
		}

		public boolean isEndLine()
		{
			if( caretLine >= lineCount - 1 )
			{
				return true;
			}
			return false;
		}
*/

/**************************************************************************
		Constraint 관련 부분
 **************************************************************************/
// 이건 다들 아시겠죠
		public void setConstraints(int constraints)
		{
			int maskedConstraints = constraints & TextField.CONSTRAINT_MASK;
			if (maskedConstraints < 0 || maskedConstraints > 4)
				throw new IllegalArgumentException();

			this.constraints = constraints;

			if(focus){
				imListener.clear();
				imListener.setTextComponent(imi);
				this.doLayout(0);
			}
		}

		public int getConstraints()
		{
			return this.constraints;
		}
/**************************************************************************
		Cursor 관련 부분
 **************************************************************************/

		public int getCaretPosition()
		{
			return caretPos;
		}
/**************************************************************************
		그밖의 메소드
 **************************************************************************/
		public void setFocus(boolean focus)
		{
			this.focus = focus;
```

```
                    if (focus)
                            imListener.setTextComponent(imi);
                    else
                            imListener.setTextComponent(null);

                    updateCaretLine();
            }

            public boolean isFocus()
            {
                    return this.focus;
            }

            public int getBytes()
            {
                    return String.valueOf(text, 0, textCount).getBytes().length;
            }
}
```

참고로 제가 정리를 잘 못한 소스라서 보시면서 조금 이상한 부분들이 나오더라도 이해해 주십시오. ㅡㅡ;
기존의 XTD-10 번 문서를 보시지 않으면 이해가 안되는 부분이 많습니다. 일단은 그것을 먼저 보신 후 에 참조하셔야 될 것 같습니다. 중복되는 부분은 삭제했습니다. 거의 중복이죠 일단 공개된 이상 자기에 맞는 소스를 가지십시오. 컨텐츠 성격, 쓰임, 용도 등..워낙 큰 Class 이기 때문에 용량도 만만치 않습니다. XTextField 클래스의 분석함으로도 여러분의 실력은 많이 향상 되실겁니다. 하나의 컴포넌트를 만든것이죠. 그것도 엄청나게 큰 컴포넌트죠. 중요하면서도…잘 이해가 안가시는 부분들은 실제적으로 값을 바꾸어 보면서 테스트해 보십시오. 언제나 고기는 낚는 법을 배우는 것이 미래의 위한 밑거름입니다.

쓸 때 없는 분석일지도 모르지만 일단은 분석했습니다. 누가 봐도 도대체 분석을 제대로 했는지 알 수가 없군요 ㅡㅡ; 솔직히 말하면..이번달에는 고민을 많이 해봤는데..프로젝트를 하나 할려다고..시간상의 압박에 쪼여서..XTextField 분석했습니다. ㅡㅡ+ ㅋㅋㅋ
다음 달에는 프로젝트 소스도 있으니..한번 해보겠습니다.

그럼 즐건 하루 되십시오..

닉네임 : 핑크레드
성격 : 다소 소심, 다소 삐짐, 다소 이중인격, 다소 수다쟁이
이메일 : pinkred@hanafos.com

추신 : 강좌를 다른 사이트에 올리시는 것은 자유입니다..단지 저한테 메일 한 통 이라도 보내주시고 올리시면 감사하겠습니다. 언제나 좋은 하루 되십시오.