핑크레드의 실전 모바일 자바 프로그래밍

바쁘거나 혹은 한가하거나 이유는 없습니다. 단지 왜 이렇게 강좌를 게을리하고 있었는지..훗훗훗
일단 후딱 이어서 강좌를 내놓아야한다는 생각이 듬뿍들어서 쓸 생각입니다.
참고로 가끔씩 즐기는 카트라이더에서 스피드의 전설을 땄습니다. 그넘의 카트라이더 스피드 전설을 딸라고 3달동안 직잭만 달렸다는..전설 따던 날 1주일동안 쉬고 있다가 30분만에 땄습니다. 어이 없다는..게임 광고는 아니고 그동안 카트를 하느라 못했다는 것도 아니고 그냥 서론이였습니다. 요즘들어서 모바일 프로그래밍을 하시는 분들이 날로 늘어가고 있는데 가끔씩 모바일 컨테츠 프로그래밍 쪽은 폭이 작고 깊이가 적어서 조금만 하면 전부들 쉽게 접근해서 이넘의 경력은 쓸모가 없어지더군요. 그나마 단말사나 혹은 플랫폼 프로그래머 정도 되야지 폭과 깊이가 있어서 인정 받더군요. 결국 이런식이 되더군요..
저도 적지않은 컨텐츠 프로그래밍 경력이 있지만 이런 생각이 문뜩 들면서 나도 심도 있게 먼가를 하지 않으면 안된다는 생각이 들더군요.
지금 입문하시는 분들에게는 별로 좋지 않게 들리겠지만 모바일 컨텐츠 1년만 만들어 보십시오. 아마 모바일은 그렇게 어렵지 않다는 것을 아실겁니다.
여튼 한번씩은 자신의 미래에 대해서 생각해 보시는 것도 좋을 듯..

전번에 이어서 후딱 소스를 만들어 보겠습니다. 별로 남은게 없는 것 같지만 아직도 전 갈길이 멀은 것 같다는...

```
public char[] getChars()

{
    char[] text = new char[list_text[lineCur].length];

    System.arraycopy(list_text[lineCur], 0,
            text, 0, text.length);

    return text;
}

public String getString()
{
    return new String(list_text[lineCur], 0, list_text[lineCur].length);
}
```

```
/**
 * 현재 선택된 라인.
 * @return
 */
public int getSelect()
{
	return lineCur;
}
```

그리 어려운 함수들은 아닙니다. 꼭 구현해야 하는 것도 아니고 필요에 구현 하시면 될 듯.

복잡한 키 이벤트 차례내요..

```
/**
 *
 * @param keyCode
 */
protected void keyPressed(int keyCode)
{
	switch(keyCode)
	{
	   // 해당 번호키를 누를경우.
	   // 룰은 현재 선택한 번호로 이동하고 난후에
	   // 키를 놓으면 움직이는 건데.
	   // 문제는 화면 밖의 번호를 누를경우 애매해진다.
	   // 화면 밖으로 이동하고 난후에 움직이기가 애매하기 때문에..
	   // 화면에 안보이는 것은 이동하지 않을지 등등의 기획 이슈가 필요하지만
	   // 본인의 생각은 안보이는 것은 보이게 한후에 이동하는 것도 멋질듯..
	   // 10개 넘는것은 Timer를 이용해야 하는데 무지 귀찮다..그냥 하자.
		case Canvas.KEY_NUM1:
		case Canvas.KEY_NUM2:
		case Canvas.KEY_NUM3:
		case Canvas.KEY_NUM4:
		case Canvas.KEY_NUM5:
		case Canvas.KEY_NUM6:
```

```
		case Canvas.KEY_NUM7:
		case Canvas.KEY_NUM8:
		case Canvas.KEY_NUM9:
			if(keyCode - Canvas.KEY_NUM1 < maxLinePerHeight)
			{
				lineCur = keyCode - Canvas.KEY_NUM1;

				canvas.serviceRepaints();
				canvas.repaint(list_x, list_y, list_width, list_height);
			}
			break;

		case Canvas.KEY_UP:
			// 현재 라인이 0보다 작으면 맨 마지막 라인으로 보낸다.
			// 어떤 리스트는 위올리다가 더이상 올라가지 않게 하는
			// 경우도 있다. 룰은 기획에 따라서..
			if(lineCur <= 0)
			{
				// 맨 마지막 라인.
				lineCur = lineMax - 1;

				// 화면의 첫번째 라인 설정 list_height 보다 큰 경우
				// 마지막 라인에서 화면에 보여주는 만큼을 뺀다.
				lineFirst = (lineMax * LIST_PER_LINEHEIGHT <=
list_height)?0:lineMax - maxLinePerHeight;
			} else
			{
				// 첫번째 라인을 하나씩 위로 올린다.
				if(lineFirst > lineCur - 1)
					lineFirst--;

				// 현재 라인도 위로 올린다.
				lineCur--;
			}
```

```
                        // 중요 체크..
                        // 현재 커서의 다음 라인이 list_height 보다 크면 list_yMove해주
                        // 어야한다.  마지막 라인이 안보이는 만큼만 위로 올린다.
                        // 이경우는 첫번째라인에서 위로 올리면 마지막라인으로 가기
                        // 때문에 생겼다.
                        if((lineCur + 1) * LIST_PER_LINEHEIGHT > list_height -
margin_top - margin_bottom)
                                gapHeightOfListAndReal = true;

                        // 일단 맨 마지막 라인을 가면 list_yMove가 true가 되서..
                        // 전체적으로 라인이 그려지는 위치를 올려서 맨 마지막
                        // 라인이 잘보이게 한다.
                        // 이런 경우 첫번째 라인이 약간만 보이게 된다. 그럴경우
                        // 다시 첫번째 라인이 제대로
                        // 보일려면 커서가 첫번째 라인이 올때까지 기다려야 한다.
                        if(gapHeightOfListAndReal)
                                gapHeightOfListAndReal = lineCur ==
lineFirst?false:true;

                        // 원래는 리스트 영역만큼 repaint()를 했는데 리스트 영역
                        // 밖으로 스크롤이 생기는 경우가
                        // 있는 경우에는 폭에서 더할수 밖에 없다...대략..
                        canvas.repaint(list_x, list_y, list_width + 10, list_height);
                        canvas.serviceRepaints();
                        break;

                case Canvas.KEY_DOWN:
                        // 현재 라인이 마지막 라인이면 첫번째라인으로 이동한다.
                        if(lineCur >= lineMax - 1)
                        {
                                lineFirst = lineCur = 0;
                        }else
                        {
```

```
                        // 첫번째 라인을 증가시킨다.
                        if(lineFirst + maxLinePerHeight - 1 <= lineCur)
                                lineFirst++;

                        // 현재 라인을 증가시킨다.
                        lineCur++;
                }

                // 위와 상동이다.
                if((lineCur + 1) * LIST_PER_LINEHEIGHT > list_height -
margin_top - margin_bottom)
                        gapHeightOfListAndReal = true;

                // 위와 상동이다.
                if(gapHeightOfListAndReal)
                        gapHeightOfListAndReal = lineCur ==
lineFirst?false:true;

                // 원래는 리스트 영역만큼 repaint()를 했는데 리스트 영역
                // 밖으로 스크롤이 생기는 경우가
                // 있는 경우에는 폭에서 더할수 밖에 없다...대략..
                // 스크롤은 대락 딴곳에서 PinkListFIeld에서 구현해서
                // 웃기지만..말이다..
                canvas.repaint(list_x, list_y, list_width + 10, list_height);
                canvas.serviceRepaints();
                break;
        }
}

/**
 * 간혹 리스트에 해당 번호를 누르면 이동하는 경우가 있는데 바로
 * 이것을 이용해서 리스너를 등록하면 가능한다.
 *
 * @param keyCode
```

```
 */
protected void keyReleased(int keyCode)
{
	switch (keyCode)
	{
		case Canvas.KEY_NUM1:
		case Canvas.KEY_NUM2:
		case Canvas.KEY_NUM3:
		case Canvas.KEY_NUM4:
		case Canvas.KEY_NUM5:
		case Canvas.KEY_NUM6:
		case Canvas.KEY_NUM7:
		case Canvas.KEY_NUM8:
		case Canvas.KEY_NUM9:
			if(keyCode - Canvas.KEY_NUM1 < maxLinePerHeight)
				keyNotify(Canvas.KEY_FIRE);
			break;
	}
}
```

매우 복잡해 보인다. 무슨 말인지도 모르겠고 설명하자면 일단 리스트에는 번호가 있다. 해당 번호를 자판에서 클릭시에 이동하는 루틴이 들어갔다. 0~9 까지이다. 두자리 숫자일 경우에는 첫번째 자리를 누른후에 타이머를 이용해서 다음 숫자까지의 시간을 체크해서 미입력이면 첫번째 자리로 이동 입력되면 두자리로 이동되는데 그러한 것은 귀찮아서 첫번째 자리만 넣었다. 어중간할 빠에는 빼버리자고 생각하시면 빼고 우리 리스트는 9자리 밑이다면 넣어도 된다. 그리고 리스트 상단과 하단에서 이동시에 맨마지막과 맨처음 리스트로 이동하게 만들어 놓았다. 이동의 편의성인데 필요 없다고 생각하시면 빼시면 됨. 그리고 gapHeightOfListAndReal 변수가 있는데 이건 화면상에서 마지막 리스트를 선택하는데 이게 어중간하게 중간에 걸쳐 있으면 살짝 위로 올려야하는데 그부분을 체크하는 변수이다. 처음 보는 메소드가 있다면 메소드의 이름으로 의미를 파악해보세요. 그래도 별 문제는 없을 듯 합니다. 결국 다 보여드리겠지만..
현재 이렇게 하면 대략적인 모습은 이해 하실겁니다. 설마 저것으로 리스트가 다 구현되지는 않습니다. 대략적인 모습입니다.
그럼 전체 소스를 한번 볼까요.

```
import javax.microedition.lcdui.Canvas;
```

```
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Graphics;

public class PinkList
{

/*
** ----------------------------------------------------------------------------
**  FINAL
** ----------------------------------------------------------------------------
*/
   // 이런 세세한것은 알아서...
   // 근데 만일 아이콘으로 할경우에는 글자 간격을 아이콘의 폭에 일정 크기만큼 늘려도 관계는 없을것이다.
   final int LIST_ITEM_WIDTH       = 16;  // 리스트의 숫자와 글자와의 간격

/*
** ----------------------------------------------------------------------------
**  FIELD
** ----------------------------------------------------------------------------
*/
   // 한줄당 라인 높이.
   int LIST_PER_LINEHEIGHT;
   int LIST_FONTHEIGHT;

   int list_x;         // List X축
   int list_y;         // List Y축
   int list_width;     // List Width
   int list_height;    // List Height

   int margin_left;    // Margin 영역 LEFT
   int margin_right;   // Margin 영역 RIGHT
   int margin_top;     // Margin 영역 TOP
   int margin_bottom;  // Margin 영역 BOTTOM
```

```
    int lineFirst;      // 현재 화면에 보이는 상단 Line
    int lineCur;        // 현재 선택된 Line
    int lineMax;        // 리스트 전체 라인 개수

    int list_anchor;        // 기타 옵션
    int maxLinePerHeight;   // LIST_HEIGHT에 들어가 최대 줄 개수

    boolean gapHeightOfListAndReal; // List에서 y축을 옮겨야 할경우.



    char[][] list_text;     // 리스트 문자 배열
    char[] list_num = {'①', '②', '③', '④', '⑤', '⑥', '⑦', '⑧', '⑨', '○'};   // List 숫자 배열

    int colorFont;     // 폰트 칼라
    int colorSelFont;  // 선택된 폰트 칼라
    int colorSelBar;   // 선택바 칼라
    int colorBack;     // 배경 색상.

    Canvas canvas;
    PinkComponentListener pinkComponentListener;

/*
** ------------------------------------------------------------------------------
**  METHOD
** ------------------------------------------------------------------------------
*/

    /**
     * 생성자
     */
    PinkList(Canvas canvas)
    {
        this.canvas = canvas;
```

```
    // 초기화.
    LIST_FONTHEIGHT = Font.getDefaultFont().getHeight();
    LIST_PER_LINEHEIGHT = LIST_FONTHEIGHT + 1;
}

/**
 * 메모리 해제.
 * 이전에는 해당 List를 그냥 넘겨주어서  레퍼런스만 받았지만
 * 이번 버전에서는 해당 List를 전부 카피해서 이용한다.
 * 그러므로 시작시마다 초기화를 해주어야 한다.
 * 장단점은 있다. 그것은 본인이 원하는 스타일로 변경하면 끝날뿐이다.
 *
 */
public void free()
{
    if(list_text != null)
    {
        for(int i = 0; i < lineMax; i++)
            list_text[i] = null;

        list_text = null;
    }
}

/**
 * 리스트의 키 이벤트를 해당 리스너를 이용하여 전달한다.
 * @param keyCode
 */
protected void keyNotify(int keyCode)
{
    if(pinkComponentListener != null)
        pinkComponentListener.keyNotify(keyCode);
}
```

```
/**
 *  최대 라인수를 구한다.
 *
 */
void setMaxLinePerHeight()
{
    // Height에 들어갈 최대 줄개수를 구합니다.
    // 뒤부분은 화면에 조금 보이더라도 줄로 인정하는 것입니다.(이부분이 줄개수의 핵심이죠)
    maxLinePerHeight = getRealHeight() / LIST_PER_LINEHEIGHT +
        ((getRealHeight() % LIST_PER_LINEHEIGHT > 0)?1:0);
}

/**
 * 리스트의 영역을 잡는다.
 *
 * @param x
 * @param y
 * @param width
 * @param height
 */
public void setBounds(int x, int y, int width, int height)
{
    list_x = x;
    list_y = y;
    list_width = width;
    list_height = height;

    setMaxLinePerHeight();
}

/**
 * margin의 크기를 잡는다.
 * margin은 setBounds 이전 혹은 다음에 설정해도 되지만 가능하면
 * 보기 좋게 setBounds 다음에 설정하도록 하자.
```

```
 *
 * @param top
 * @param right
 * @param bottom
 * @param left
 */
public void setMargin(int top, int right, int bottom, int left)
{
    margin_top      = top;
    margin_right    = right;
    margin_bottom   = bottom;
    margin_left     = left;

    setMaxLinePerHeight();
}

/**
 * 리스트의 라인 높이를 조절한다.
 * 기본은 기본폰트 + 1 로 잡혀있다.
 *
 * @param height
 */
public void setListPerLineHeight(int height)
{
    LIST_PER_LINEHEIGHT = height;

    setMaxLinePerHeight();
}

/**
 * 리스너 등록 리스트의 이벤트를 캔버스로 날려주기 위해서 쓴다.
 * 예를 들어 1, 2, 3 등등의 번호 이벤트를 위해서.
 * @param aTimerListener
 */
```

```
public void setComponentListener(PinkComponentListener xComponentListener)
{

    this.pinkComponentListener = null;
    this.pinkComponentListener = xComponentListener;
}

/**
 * 리스트를 set한다.
 * @param list
 * @param count
 * @param anchor
 */
public void setChars(char[][] list, int count, int anchor)
{
    int length = 0;

    // 예외처리
    if(list == null || list.length < count || count <= 0)
        throw new ArrayIndexOutOfBoundsException();

    free();

    // 변수 초기화
    lineFirst = 0;
    lineCur = 0;
    lineMax = count;
    gapHeightOfListAndReal = false;
    this.list_anchor = anchor;

    // 여기서 새로운 타입이 바뀔때마다 생성 할것인지.
    // 아니면 미리 전부 생성해 놓고 넘겨만 줄것인지는 본인들이 선택.
    // 나도 모르겠삼.
    list_text = new char[lineMax][];
```

```
    for(int i = 0; i < lineMax; i++)
    {
        length = list[i].length;
        list_text[i] = new char[length];

        System.arraycopy(list[i], 0, list_text[i], 0, length);
    }
}


/**
 * 가능하면 String를 쓰지 말자.
 * @param list
 * @param count
 * @param anchor
 */
public void setString(String[] list, int count, int anchor)
{
    // 예외처리
    if(list == null || list.length < count || count <= 0)
        throw new ArrayIndexOutOfBoundsException();

    free();

    // 변수 초기화
    lineFirst = 0;
    lineCur = 0;
    lineMax = count;
    gapHeightOfListAndReal = false;
    this.list_anchor = anchor;

    list_text = new char[lineMax][];
```

```
    for(int i = 0; i < lineMax; i++)
        list_text[i] = list[i].toCharArray();
}

/**
 * 칼라를 변경한다. 근데 이것외에도변경해야할 칼라도 있고 쓸모 없는 칼라도
 * 있다. 본인이 바꾸도록 하자.
 *
 * @param back  배경
 * @param font  글자
 * @param selFont   선택된 글자
 * @param selBar    선택바
 */
public void setColor(int back, int font, int selFont, int selBar)
{
    colorBack = back;
    colorFont = font;
    colorSelFont = selFont;
    colorSelBar = selBar;
}

/**
 * 화면상의 마지막 라인.
 *
 * @return
 */
int getLastLine()
{
    // List의 마지막 라인을 찾는다.
    // 기본적으로 전체 라인 개수가 maxLinePerHeight 작거나 같으면 마지막 라인이 전체라인이 되고
    // 그렇지 않으면 첫번째라인에서 maxLinePerHeight 만큼 더해서 마지막 라인을 계산한다.
    if(lineMax <= maxLinePerHeight || lineMax <= lineFirst + maxLinePerHeight)
        return lineMax;
    else
```

```
        return lineFirst + maxLinePerHeight;
}

int getRealX()
{
    return list_x + margin_left;
}

int getRealY()
{
    int result = list_y + margin_top;

    if(gapHeightOfListAndReal)
        result -= getGapHeightOfListAndReal();

    return result;
}

/**
 * list의 높이와 화면상의 전체 라인 X 폰트의 높이를 구해서 해당 차이..
 * @return
 */
int getGapHeightOfListAndReal()
{
    return maxLinePerHeight * LIST_PER_LINEHEIGHT - list_height;
}

int getMaxLinePerHeight()
{
    return LIST_PER_LINEHEIGHT;
}

int getTextY(int line)
{
```

```
    return getRealY() + LIST_PER_LINEHEIGHT * line;
}

int getSelLine()
{
    return lineCur - lineFirst;
}

int getGapFontHeightOfLine()
{
    return (LIST_PER_LINEHEIGHT - LIST_FONTHEIGHT) >> 1;
}

int getRealHeight()
{
    return list_height - margin_top - margin_bottom;
}

int getRealWidth()
{
    return list_width - margin_left - margin_right;
}

int getWidth()
{
    return list_width;
}

int getHeight()
{
    return list_height;
}

/**
```

```
 * 현재 선택된 라인.
 * @return
 */
public int getSelect()
{
    return lineCur;
}

/**
 * 현재 선택된 문자.
 * @return
 */
public char[] getChars()
{
    char[] result = new char[list_text[lineCur].length];

    System.arraycopy(list_text[lineCur], 0,
            result, 0, result.length);

    return result;
}

/**
 * 언제나 String은 가급적 쓰지 말자.
 * @return
 */
public String getString()
{
    return new String(list_text[lineCur], 0, list_text[lineCur].length);
}

/**
 * 화면에 그린다.
 *
```

```
 * @param g
 */
public void paint(Graphics g)
{
    int line = lineFirst;
    int k = 0;
    while(line < getLastLine())
    {
        // 현재 선택된 라인일경우.
        if(k == getSelLine())
        {
            drawBar(g, list_x, getTextY(k));
            drawList(g, getRealX(), getTextY(k++), line++, colorSelFont);
        }else
            drawList(g, getRealX(), getTextY(k++), line++, colorFont);
    }
}

/**
 * 현재 선택된 것을 보여준다.
 * @param g
 * @param x
 * @param y
 */
protected void drawBar(Graphics g, int x, int y)
{
    g.setColor(colorSelBar);
    g.fillRect(x, y, list_width, LIST_PER_LINEHEIGHT);
}

/**
 * 해당 리스트를 그린다.
 *
 * @param g
```

```
 * @param x
 * @param y
 * @param line
 * @param color
 */
void drawList(Graphics g, int x, int y, int line, int color)
{
    int text_x = x;
    int text_y = y + ((LIST_PER_LINEHEIGHT - LIST_FONTHEIGHT) >> 1);

    g.setColor(color);

    // 숫자 그리기
    if(list_num.length - 1 > line)
        g.drawChar(list_num[line], text_x, text_y, Graphics.TOP|Graphics.LEFT);
    else
        g.drawChar(list_num[9], text_x, text_y, Graphics.TOP|Graphics.LEFT);

    text_x += LIST_ITEM_WIDTH;

    //리스트 목록 그리기
    g.drawChars(list_text[line], 0, list_text[line].length, text_x, text_y, Graphics.TOP|Graphics.LEFT);
}

/**
 *
 * @param keyCode
 */
protected void keyPressed(int keyCode)
{
    switch(keyCode)
    {
        // 해당 번호키를 누를경우.
        // 룰은 현재 선택한 번호로 이동하고 난후에
```

```
// 키를 놓으면 움직이는 건데.
// 문제는 화면 밖의 번호를 누를경우 애매해진다.
// 화면 밖으로 이동하고 난후에 움직이기가 애매하기 때문에..
// 화면에 안보이는 것은 이동하지 않을지 등등의 기획 이슈가 필요하지만
// 본인의 생각은 안보이는 것은 보이게 한후에 이동하는 것도 멋질듯..
// 10개 넘는것은 Timer를 이용해야 하는데 무지 귀찮자..그냥 하자.
case Canvas.KEY_NUM1:
case Canvas.KEY_NUM2:
case Canvas.KEY_NUM3:
case Canvas.KEY_NUM4:
case Canvas.KEY_NUM5:
case Canvas.KEY_NUM6:
case Canvas.KEY_NUM7:
case Canvas.KEY_NUM8:
case Canvas.KEY_NUM9:
    if(keyCode - Canvas.KEY_NUM1 < maxLinePerHeight)
    {
        lineCur = keyCode - Canvas.KEY_NUM1;

        canvas.repaint(list_x, list_y, list_width, list_height);
        canvas.serviceRepaints();
    }
    break;

case Canvas.KEY_UP:
    // 현재 라인이 0보다 작으면...맨 마지막 라인으로 보낸다.
    // 어떤 리스트는 위올리다가 더이상 올라가지 않게 하는 경우도 있다.
    if(lineCur <= 0)
    {
        // 맨 마지막 라인.
        lineCur = lineMax - 1;

        // 화면의 첫번째 라인 설정
        // list_height 보다 큰 경우.
```

```
            // 마지막 라인에서 화면에 보여주는 만큼을 뺀다.
            lineFirst = (lineMax * LIST_PER_LINEHEIGHT <= list_height)?0:lineMax - maxLinePerHeight;
        } else
        {
            // 첫번째 라인을 하나씩 위로 올린다.
            if(lineFirst > lineCur - 1)
                lineFirst--;

            // 현재 라인도 위로 올린다.
            lineCur--;
        }

        // 중요 체크..

        // 현재 커서의 다음 라인이 list_height 보다 크면 list_yMove해주어야한다.
        // 마지막 라인이 안보이는 만큼만 위로 올린다.
        // 이경우는 첫번째라인에서 위로 올리면 마지막라인으로 가기 때문에 생겼다.
        if((lineCur + 1) * LIST_PER_LINEHEIGHT > list_height - margin_top - margin_bottom)
            gapHeightOfListAndReal = true;

        // 일단 맨 마지막 라인을 가면 list_yMove가 true가 되서..
        // 전체적으로 라인이 그려지는 위치를 올려서 맨 마지막 라인이 잘보이게 한다.
        // 이런 경우 첫번째 라인이 약간만 보이게 된다. 그럴경우 다시 첫번째 라인이 제대로
        // 보일려면 커서가 첫번째 라인이 올때까지 기다려야 한다.
        if(gapHeightOfListAndReal)
            gapHeightOfListAndReal = lineCur == lineFirst?false:true;

        // 원래는 리스트 영역만큼 repaint()를 했는데 리스트 영역 밖으로 스크롤이 생기는 경우가
        // 있는 경우에는 폭에서 더할수 밖에 없다...대략..
        canvas.repaint(list_x, list_y, list_width + 10, list_height);
        canvas.serviceRepaints();
        break;

    case Canvas.KEY_DOWN:
```

```
            // 현재 라인이 마지막 라인이면 첫번째라인으로 이동한다.
            if(lineCur >= lineMax - 1)
            {
                lineFirst = lineCur = 0;
            }else
            {
                // 첫번째 라인을 증가시킨다.
                if(lineFirst + maxLinePerHeight - 1 <= lineCur)
                    lineFirst++;

                // 현재 라인을 증가시킨다.
                lineCur++;
            }

            // 위와 상동이다.
            if((lineCur + 1) * LIST_PER_LINEHEIGHT > list_height - margin_top - margin_bottom)
                gapHeightOfListAndReal = true;

            // 위와 상동이다.
            if(gapHeightOfListAndReal)
                gapHeightOfListAndReal = lineCur == lineFirst?false:true;

            // 원래는 리스트 영역만큼 repaint()를 했는데 리스트 영역 밖으로 스크롤이 생기는 경우가
            // 있는 경우에는 폭에서 더할수 밖에 없다...대략..
            // 스크롤은 대략 딴곳에서 XListFIeld에서 구현해서 웃기지만..말이다..
            canvas.repaint(list_x, list_y, list_width + 10, list_height);
            canvas.serviceRepaints();
            break;
        }
    }

    /**
     * 간혹 리스트에 해당 번호를 누르면 이동하는 경우가 있는데 바로
     * 이것을 이용해서 리스너를 등록하면 가능한다.
```

```
     *
     * @param keyCode
     */
    protected void keyReleased(int keyCode)
    {
        switch (keyCode)
        {
            case Canvas.KEY_NUM1:
            case Canvas.KEY_NUM2:
            case Canvas.KEY_NUM3:
            case Canvas.KEY_NUM4:
            case Canvas.KEY_NUM5:
            case Canvas.KEY_NUM6:
            case Canvas.KEY_NUM7:
            case Canvas.KEY_NUM8:
            case Canvas.KEY_NUM9:
                if(keyCode - Canvas.KEY_NUM1 < maxLinePerHeight)
                    keyNotify(Canvas.KEY_FIRE);
                break;
        }
    }
}
```

이거 다 칠라면 죽을지도 모릅니다. ㅎㅎㅎ 일단은 이 소스는 이렇게만 공유해 드리고요.
다음 소스를 파일까지드리겠습니다. 다음 소스란 무언인가 위의 리스트의 문제점은 바로 한가지 경우에만 사용가능한 버전이라는 것입니다. 일반적으로 컨텐츠에는 여러가지 리스트가 존재하게 되는데 그럼 결국 리스트를 여러 개 만들수는 없으니 해당 타입에 맞추어서 리스트의 모습을 변경시켜주면 쉽지 않겠습니까? 그래서 타입을 넣는 방식이 추가되면서 새로운 구조로 바뀌게 됩니다. 해당 타입은 anchor 로 setChars에서 받게 됩니다. 그리고 첫번째 시간에 보여드린 interface를 이용해서 구현하게 됩니다. 그렇다면 바뀌는 부분은 얼마 없습니다.

```
public final static int EMPTY      = 1;

public final static int DEFAULT = 2;

public final static int ICON       = 3; // 숫자 대신 아이콘을 경우에는 앞에 한바이트에 아이콘의 정보를 넣는다.
```

```
// 인터페이스를 만듭니다. 필요한 메소드들은 추가하셔도 되고 빼셔도 됩니다.
interface XListType
{
    abstract int getX();
    abstract int getY();
    abstract int getWidth();
    abstract char[] getChars();

    abstract void drawBar(Graphics g, int x, int y);
    abstract void drawList(Graphics g, int x, int y, int line, int color);
}

/**
 * 리스트를 set한다.
 * @param list
 * @param count
 * @param anchor
 */
public void setChars(char[][] list, int count, int anchor)
{
		int length = 0;

		// 예외처리
		if(list == null || list.length < count || count <= 0)
			throw new ArrayIndexOutOfBoundsException();

		free();

		// 변수 초기화
		lineFirst = 0;
		lineCur = 0;
		lineMax = count;
		gapHeightOfListAndReal = false;
		this.list_anchor = anchor;
```

```
        xListType = null;

    // 여기서 새로운 타입이 바뀔때마다 생성 할것인지.
    // 아니면 미리 전부 생성해 놓고 넘겨만 줄것인지는 본인들이 선택.
    // 나도 모르겠삼. 각각의 타입인데 그건 기획대로..해주세요.
    switch(anchor)
    {
        case EMPTY:
            xListType = new XListTypeOfEmpty();
            break;

        case DEFAULT:
            xListType = new XListTypeOfDefault();
            break;

        case ICON:
            xListType = new XListTypeOfIcon();
            break;
    }

            list_text = new char[lineMax][];

            for(int i = 0; i < lineMax; i++)
            {
                        length = list[i].length;
                        list_text[i] = new char[length];

                        System.arraycopy(list[i], 0, list_text[i], 0, length);
            }
}

/**
    * 화면에 그린다.
    *
```

```
     * @param g
     */
    public void paint(Graphics g)
    {
        int line = lineFirst;
        int k = 0;
        while(line < getLastLine())
        {
            // 현재 선택된 라인일경우.
            if(k == getSelLine())
            {
                xListType.drawBar(g, list_x, getTextY(k));
                xListType.drawList(g, getRealX(), getTextY(k++), line++, colorSelFont);
            }else
                xListType.drawList(g, getRealX(), getTextY(k++), line++, colorFont);
        }
    }

/*
** ------------------------------------------------------------------------------
**  XList Type
** ------------------------------------------------------------------------------
*/

    class XListTypeOfEmpty implements XListType
    {
        public int getX()
        {
            return getRealX();
        }

        public int getY()
        {
            return getTextY(getSelLine()) + getGapFontHeightOfLine();
```

```
    }

    public int getWidth()
    {
        return getRealWidth();
    }

    public char[] getChars()
    {
        char[] text = new char[list_text[lineCur].length];

        System.arraycopy(list_text[lineCur], 0,
                text, 0, text.length);

        return text;
    }

    public void drawBar(Graphics g, int x, int y)
    {
        // 선택 바 그리기.
        g.setColor(colorSelBar);
        g.fillRect(x, y, list_width, LIST_PER_LINEHEIGHT);
    }

    public void drawList(Graphics g, int x, int y, int line, int color)
    {
        g.setColor(color);

        //리스트 목록 그리기
        g.drawChars(list_text[line], 0, list_text[line].length,
                x, getTextY(line), Graphics.TOP|Graphics.LEFT);
    }
}
```

```
class XListTypeOfDefault implements XListType
{
    public int getX()
    {
        return getRealX() + LIST_ITEM_WIDTH;
    }

    public int getY()
    {
        return getTextY(getSelLine()) + getGapFontHeightOfLine();
    }

    public int getWidth()
    {
        return getRealWidth() - LIST_ITEM_WIDTH;
    }

    public char[] getChars()
    {
        char[] text = new char[list_text[lineCur].length];

        System.arraycopy(list_text[lineCur], 0,
                text, 0, text.length);

        return text;
    }

    public void drawBar(Graphics g, int x, int y)
    {
        // 선택 바 그리기.
        g.setColor(colorSelBar);
        g.fillRect(x, y, list_width, LIST_PER_LINEHEIGHT);
```

```
    }

    public void drawList(Graphics g, int x, int y, int line, int color)
    {
        g.setColor(color);

        // 숫자 그리기
        if(list_num.length - 1 > line)
        {
            g.drawChar(list_num[line],
                    x, getTextY(line), Graphics.TOP|Graphics.LEFT);
        }else
        {
            g.drawChar(list_num[9],
                    x, getTextY(line), Graphics.TOP|Graphics.LEFT);
        }

        //리스트 목록 그리기
        g.drawChars(list_text[line], 0, list_text[line].length,
                x + LIST_ITEM_WIDTH, getTextY(line), Graphics.TOP|Graphics.LEFT);

    }
}

class XListTypeOfIcon implements XListType
{
    public int getX()
    {
        return getRealX() + LIST_ITEM_WIDTH;
    }

    public int getY()
    {
        return getTextY(getSelLine()) + getGapFontHeightOfLine();
```

```
}

public int getWidth()
{
    return getRealWidth() - LIST_ITEM_WIDTH;
}

public char[] getChars()
{
    char[] text = new char[list_text[lineCur].length - 1];

    System.arraycopy(list_text[lineCur], 1,
            text, 0, text.length);

    return text;
}

public void drawBar(Graphics g, int x, int y)
{
    // 선택 바 그리기.
    g.setColor(colorSelBar);
    g.fillRect(x, y, list_width, LIST_PER_LINEHEIGHT);
}

public void drawList(Graphics g, int x, int y, int line, int color)
{
    g.setColor(color);

    // 아이콘 그리기.
    ////////////////////////////////////////////////////////

    //리스트 목록 그리기
    g.drawChars(list_text[line], 0, list_text[line].length,
            x + LIST_PER_LINEHEIGHT, getTextY(line), Graphics.TOP|Graphics.LEFT);
```

```
    }
  }
}
```

이렇게 됨으로 중복되는 drawbar(), drawList()는 빠지고 각각의 값에 타입에 따라서 내부적으로 구현되기 때문에 switch가 없어짐으로써 속도 향상에도 많은 역할을 합니다. 또한 구조적으로 수정도 쉽고 있어보이죠. 내용이 길어서 복잡해 보이시기도 하시겠지만 직접 치시면서 이해하시면 쉬우실듯도 하고 혹은 이론만 생각하시고 직접 본인이 새로 구현하셔도 도움이 되실겁니다. 여태까지 한 것이 PinkList였습니다. PinkListField, PinkPopup은 위의 PinkList를 기본으로 해서 스크롤과 티커가 생기면 ListField로 그리고 이미지와 위치가 수정되면 Popup로 됩니다. 상속형태이기 때문에 그리 문제도 안되고 어려운 점은 없으실 듯 합니다. ListField, Popup의 소스는 다음에 공개하도록 하겠습니다. 한꺼번에 너무 많이 드리면 머리가 복잡해 집니다.

닉네임 : 핑크레드
성격 : 왕소심, 왕삐짐, 이중인격, 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.