핑크레드의 실전 모바일 자바(SK-VM) 프로그래밍

안녕하세요 핑크레드 입니다.
이번에 프로젝트가 빡쌔게 끝나서..다행이 시간이 좀 생겼습니다. ㅎㅎㅎ
그래서 이번 프로젝트에서 익힌 것을 알려드리려고 합니다.

이전과 마찬가지로 새로운 것을 안다는 것은 참 기쁘죠..
예전에 제가 PinkList를 알려드린적이 있습니다. 그런데 이번 프로젝트를 하면서 새로운 List를 만들어야 하는 이슈가 발생해서 다시 만들었는데 예전보다 훨씬 좋은 List가 나오고 말았습니다. ㅎㅎㅎ 기분이 째지고 말았죠. 근디..이것을 여러분들에게 알려드리려고 하니.. 저의 노하우가 바로 공개되는 것 같아서 가끔은 나의 정보가 빠져나가는 구나 하는 생각도 들기도 합니다. 어떤 분이 이런말을 하셨습니다. 지식 경영이라고 해서 모든 사원들의 지식을 DB로 만들어서 누구든 시작 시 같은 지식을 공유하게 되는 거죠..사원의 평등화가 생기면서 이제는 누가 더 지식을 잘 정리하는지 지식을 더 많이 가지고 있는지가 관건이라고 이렇게 되면 노하우는 지식으로 빠지면서 더 이상 나만의 것이 아니지요.. 이런 생각을 하면 아..나는 어쩌나..하는 생각도 들기도 합니다. 그래서 가끔은 강좌도 텀을 두고 써야할지하는 생각도 듭니다. 제가 생각한 것을 한 6개월 정도 써먹고 알려주기 같은거요..ㅎㅎㅎ

근데 어차피 누구나 보면 다 할수 있는 거라서..저도 특별히 손해 본다고는 생각하지는 않습니다. 이런 강좌를 쓰면 저 또한 해당 기술들을 정리하고 손보기 때문에 더 좋다고 생각합니다. 앗..말이 샜습니다. 다시 본론으로..뾰로롱..

이번에는 강좌를 잘 만들려고 합니다. 이론부터 충실히..제목도 이글만 읽으면 누구나 List를 만들수있다라던가..나만큼 한다..일주일만..등등.

제가 강좌를 쓰면서 이런 질문을 받은적이 있습니다. 구현시에 일일이 전부 생각하면서 알고리즘을 짜고 하시는지..ㅎㅎㅎ 모바일을 지원되는 API는 아주 기초적인 것으로써 필요한 모든 것을 만들어야 합니다. 라고 답변했습니다. 당근 알고리즘 들어가죠..그래서 List를 만들면서..고려했던것들 필요한 것들을 처음부터 설명드리겠습니다.

1. List 처음 생각.

A. 일반적으로 리스트를 모바일에서 보여줄 때 여러가지 방식의 룰들이 있을것입니다. 어떤것을 선택하느냐가 관건이죠. 일단 그림을 보시면서..ㅎㅎㅎ 왼쪽의 그림을 보면 기본적인 스타일입니다. List는 언제나 꽉차게 나오는 것입니다. 아래에 만일 조금의 여백이 있다면 줄당 간격을 두어서 약간 벌어지게 해서 언제나 한 화면에 리스트들이 보이는 형식입니다. 제가 처음에 만들었던 리스트 입니다. 키를 아래로 내리면 다음 라인이 보이는 것이지요. 포인트는 한 화면에 꽉찬 리스트 라인이죠.

B. 이번에 제가 새로 구성한 리스트 입니다. 그림은 딴 것을 얻어왔찌만.. 일단 위의것과 무엇이 달려져 보이시는지요.. 잘보시면 위의 리스트는 꽉찬 언제나 꽉채워야하는 부담이 생깁니다. 리스트 줄간격도 고려해야하고 하지만 새로 구성한 리스트는 줄간격을 고려 안해도 되고 오히려 더 자연스러워 보이는 리스트가 구성이 되는 것이지요. 진정한 리스트라고 할수 있죠라고 본인만 생각하고 있습니다. 화면의 크기에 관계 없이 리스트가 보이는 것이죠 ..작으면 작은 부분만큼 짤리는 거죠.

프로그램시에는 일단 무엇을 만들것인지 A, B형식 중 어떤 것을 할것인지를 결정해야 합니다. 여기서는 B형식으로 하기로 하겠습니다. B형식이 머 더 쉽지 않냐라고 하시는 분들도 있겠지만..그럼 할말 없습니다. 하지만..약간은 어렵습니다. 기존것보다는요..ㅎㅎ

2. 그럼 무엇무엇들이 필요할까요//??????

이제는 해당 리스트를 구성하기에 필요한 메소드들을 생각해 봅시다.

많은 것은 필요없습니다. 가장 기본적인 것부터 생각해 봅시다.

A. List의 영역을 구성하는 것이 필요하겠지요

public void setBounds()

B. List의 내용을 넣는 것이 필요하겠지요.

public void setChars(char[][] text, int num, int anchor)

public void setString(String[] text, int num, int anchor)

C. List의 내용을 화면에 그리는 것이 필요하겠지요.

public void paint(Graphics g)

public void drawList(Graphics g)

public void drawScroll(Graphics g, int tot_num, int sel_num)

D. List를 위아래로 움직이는 것이 필요하겠지요.

public void keyPressed(int keyCode)

public void keyReleased(int keyCode)

E. List를 선택한 값을 얻어오는 것들이 필요하겠지요.

public int getSelect()

public char[] getChars()

이것이 가장 기본이 되는 것들입니다. 이렇게 구성이 된다면..끝난것이지요.
이제 할일을 해당 메소드의 내용을 채우는 것입니다. 여기까지 읽으셨다면 한번 본인이 직접 메소드에 코드를ㄹ 넣어 보는 것도 좋으리라 생각됩니다. 앞으로 제가 제공하는 메소드의 내용과 비교해서 자신의 것과 제가 제공한것과 어느것이 더 좋았는지 혹은 배울점은 없는지 아니면... PinkList에서 고칠 것은 없는지..이러한 것들이 여러분들에게는 많은 도움이 됩니다.

3. 메소드의 내용을 알차게 채워 봅시다.

A. public void setBounds()

List의 영역을 표시하는 것입니다. 그러므로 List의 영역을 표시하는 변수들이 필요하죠. List영역을 하기 위해서는 한 줄당 높이와 한 화면에 보여줄 전체라인 개수등이 필요합니다.

```
public class PinkList
{
        // 한줄당 라인 높이..일반적으로 폰트 크기 보다 1정도 크게 하는데.
        // 이런 세세한것은 알아서...
         public final static int LIST_PER_LINEHEIGHT    = 15;

         int LIST_X;                    // List X축
         int LIST_Y;                    // List Y축
         int LIST_WIDTH;                // List Width
         int LIST_HEIGHT;               // List Height
```

```
        int maxPerLine              // LIST_HEIGHT에 들어가 최대 줄 개수

/*
** ------------------------------------------------------------------------
**  Function:
**          setBounds
**
**  Description:
**          리스트의 영역 설정
**
**  Input:
**      x : x축
**      y : y축
**      width : 폭
**      height : 넓이
**
**  Return value:
**
** ------------------------------------------------------------------------
*/
        public void setBounds(int x, int y, int width, int height)
        {
                LIST_X = x;
                LIST_Y = y;
                LIST_WIDTH = width;
                LIST_HEIGHT = height;

                // Height에 들어갈 최대 줄개수를 구합니다.
                // 뒤부분은 화면에 조금 보이더라도 줄로 인정하는 것입니다.
                // (이부분이 줄개수의 핵심이죠)
                maxPerLine = LIST_HEIGHT / LIST_PER_LINEHEIGHT +
                  ((LIST_HEIGHT % LIST_PER_LINEHEIGHT > 0)?1:0);
        }
```

```
        public void setChars(char[][] text, int num, int anchor){}
        public void setString(String[] text, int num, int anchor){}
        public void paint(Graphics g){}
        public void drawList(Graphics g){}
        public void drawScroll(Graphics g, int tot_num, int sel_num){}
        public void keyPressed(int keyCode){}
        public void keyReleased(int keyCode){}
        public int getSelect(){}
        public char[] getChars(){}
}
```

B. public void setChars(char[][] text, int num, int anchor)
public void setString(String[] text, int num, int anchor)

두 가지를 전부 지원하겠습니다. 가능하면 메모리 관리등의 효율성으로 고려햐면 char으로 하는 것이 좋지만 편의성을 제공하여 String도 넣었습니다. 이번 버전은 소스에도 설명되어 있듯이 리스트 내용을 복사해서 사용합니다. 기존에는 레퍼런스를 직접 받았지만.. 효율성 및 기타 등등의 문제는 고치셔서 사용하셔도 무난합니다.

```
/*
** ------------------------------------------------------------------------
**  Function:
**          setChars
**
**  Description:
**          리스트 배열 문자 세팅
**
**  Input:
**      list : 리스트 배열
**      count : 배열 개수
**      anchor : 기타 옵션(사용하지 않는다 그냥 옵션은 옵션일 뿐이다.)
**
**  Return value:
**
** ------------------------------------------------------------------------
*/
```

```
public void setChars(char[][] list, int count, int anchor)
{
	int length = 0;
	int i = 0;

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

	// list_text 초기화
	// 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.
	if(list_text != null)
	{
		for(i = 0; i < maxLine; i++)
			list_text[i] = null;

		list_text = null;
	}

	// 변수 초기화
	firstLine = 0;
	curLine = 0;
	maxLine = count;
	list_anchor = anchor;

	list_text = new char[maxLine][];

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

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


    public void setString(String[] list, int count, int anchor)
    {
            int i = 0;

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

            if(list_text != null)
            {
                    for(i = 0; i < maxLine; i++)
                            list_text[i] = null;

                    list_text = null;
            }

            // 변수 초기화
            firstLine = 0;
            curLine = 0;
            maxLine = count;
            list_anchor = anchor;

            list_text = new char[maxLine][];

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

C. public void paint(Graphics g)
public void drawList(Graphics g)
public void drawScroll(Graphics g, int tot_num, int sel_num)

이제 한번 그림을 그려 봅시다. 제일 힘들면서 많은 부분을 비중이 큰 부분입니다.
기본적이면서도 어떻게 구현했는지에 따라서 효율성이 있을지..모르겠네요..ㅎㅎㅎ
그냥 화면에 보이지만..이러한 것들이 어떻게 그려지는지가 하나의 알고리즘이 되는 것이지요. 리스트의 형태는 많은 부분에서 사용하기 때문에 도움이 되었으면 합니다. 참고로 이번 리스트는 제가 예전 강좌에서 선보인 이벤트 구조로 만들었습니다. 약간 이상하시더라도 이해를 부탁드리고 잘 이해가 안가시는 분들은 이전 강좌를 보시면 되겠습니다.

```
import java.io.*;
import java.util.*;
import javax.microedition.lcdui.*;

import com.xce.lcdui.*;
import com.xce.io.*;
import com.skt.m.*;

public class PinkList
{

/*
** ----------------------------------------------------------------------------
**        Define 선언
** ----------------------------------------------------------------------------
*/

        // 한줄당 라인 높이..일반적으로 폰트 크기 보다 1정도 크게 하는데.
        // 이런 세세한것은 알아서...
        public final static int LIST_PER_LINEHEIGHT    = 15;
        public final static int LIST_ITEM_WIDTH        = 15;    // 리스트의 숫자와 글자와의 간격

/*
** ----------------------------------------------------------------------------
```

```
**      변수 선언
** ------------------------------------------------------------------------
*/

        int LIST_X;                 // List X축
        int LIST_Y;                 // List Y축
        int LIST_WIDTH;             // List Width
        int LIST_HEIGHT;            // List Height


        int firstLine;              // 현재 화면에 보이는 상단 Line
        int curLine;                // 현재 선택된 Line
        int maxLine;                // 리스트 전체 라인 개수

        int maxPerLine;             // LIST_HEIGHT에 들어가 최대 줄 개수

        int list_anchor;            // 기타 옵션
        boolean list_yMove;         // 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 fontColor = 0xff0000;           // 폰트 칼라
        int selfontColor = 0xff0000;        // 선택된 폰트 칼라
        int barColor = 0x000000 ;           // 선택바 칼라


        // 스크롤 관련
        public final static int SCROLL_WIDTH        = 6;    // 스크롤의 배경 폭
        public final static int SCROLLBAR_WIDTH     = 6;    // 스크롤 바의 폭
        public final static int SCROLLBAR_HEIGHT    = 29;   // 스크롤 바의 높이


        PinkCanvas pinkCanvas;
```

```
/*
** ---------------------------------------------------------------------------
**  Function:
**          XList
**
**  Description:
**          생성자
**
**  Input:
**
**  Return value:
**
** ---------------------------------------------------------------------------
*/

        PinkList(PinkCanvas pinkCanvas)
        {
                this.pinkCanvas = pinkCanvas;
        }




/*
** ---------------------------------------------------------------------------
**  Function:
**          setBounds
**
**  Description:
**          리스트의 영역 설정
**
**  Input:
**      x : x축
**      y : y축
**      width : 폭
**      height : 넓이
**
```

```
**   Return value:
**
** ------------------------------------------------------------------------
*/

        public void setBounds(int x, int y, int width, int height)
        {
                LIST_X = x;
                LIST_Y = y;
                LIST_WIDTH = width;
                LIST_HEIGHT = height;

                // Height에 들어갈 최대 줄개수를 구합니다.
                // 뒤부분은 화면에 조금 보이더라도 줄로 인정하는 것입니다.
                // (이부분이 줄개수의 핵심이죠)
                maxPerLine = LIST_HEIGHT / LIST_PER_LINEHEIGHT +
                  ((LIST_HEIGHT % LIST_PER_LINEHEIGHT > 0)?1:0);
        }


/*
** ------------------------------------------------------------------------
**   Function:
**           setChars
**
**   Description:
**           리스트 배열 문자 세팅
**
**   Input:
**       list : 리스트 배열
**       count : 배열 개수
**       anchor : 기타 옵션(사용하지 않는다 그냥 옵션은 옵션일 뿐이다.)
**
**   Return value:
**
** ------------------------------------------------------------------------
```

```
*/
	public void setChars(char[][] list, int count, int anchor)
	{
		int length = 0;
		int i = 0;

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

		// list_text 초기화
		// 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.
		if(list_text != null)
		{
			for(i = 0; i < maxLine; i++)
				list_text[i] = null;

			list_text = null;
		}

		// 변수 초기화
		firstLine = 0;
		curLine = 0;
		maxLine = count;
		list_anchor = anchor;

		list_text = new char[maxLine][];

		for(i = 0; i < maxLine; i++)
		{
			length = list[i].length;
```

```
                        list_text[i] = new char[length];

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


        public void setString(String[] list, int count, int anchor)
        {
                int i = 0;

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

                if(list_text != null)
                {
                        for(i = 0; i < maxLine; i++)
                                list_text[i] = null;

                        list_text = null;
                }

                // 변수 초기화
                firstLine = 0;
                curLine = 0;
                maxLine = count;
                list_anchor = anchor;

                list_text = new char[maxLine][];

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

```
/*
** ------------------------------------------------------------------------
**  Function:
**          pinkList_EventProc
**
**  Description:
**          이벤트 프로시져.
**
**  Input:
**      status : 현재 이벤트
**      g : Graphics 객체
**      keyCode : 키값.
**
**  Return value:
**
** ------------------------------------------------------------------------
*/

        public void pinkList_EventProc(int status, Graphics g, int keyCode)
        {
                switch(status)
                {
                        case PinkCanvas.EVT_PAINT:
                                drawList(g);
                                break;

                        case PinkCanvas.EVT_KEY_PRESSED:
                                switch(keyCode)
                                {
                                        case Canvas.KEY_UP:
                                                break;

                                        case Canvas.KEY_DOWN:
                                                break;
```

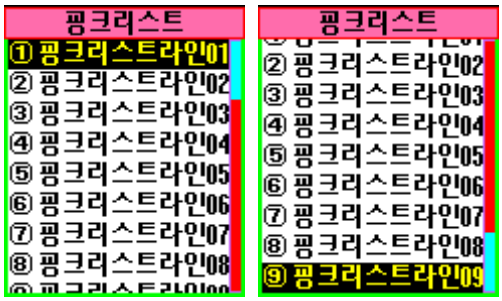
```
                                        }
                                        break;

                                case PinkCanvas.EVT_KEY_RELEASED:
                                        break;
                        }
                }


/*
** ------------------------------------------------------------------------
**  Function:
**          drawList
**
**  Description:
**          List 를 그린다.
**
**  Input:
**      g : Graphics 객체
**
**  Return value:
**
** ------------------------------------------------------------------------
*/
        public void drawList(Graphics g)
        {
                // List의 마지막 라인을 찾는다.
                // 기본적으로 전체 라인 개수가 maxPerLine 작거나 같으면 마지막 라인이
                // 전체라인이 되고
                // 그렇지 않으면 첫번째라인에서 maxPerLine 만큼 더해서 마지막 라인을
                // 계산한다.
                int lastLine = (maxLine <= maxPerLine)?maxLine:
                        ((firstLine + maxPerLine   > maxLine)?maxLine:firstLine + maxPerLine);

                int i, k;
```

```
int list_y  = LIST_Y;            // Y축이 변할 수 있다.
int list_width = LIST_WIDTH; // 폭이 변할 수 있다.

int text_x;           // 글자를 그리는 X축
int text_y;           // 글자를 그리는 Y축

// Scroll 를 그릴것인지 안닌지.
boolean bDrawScroll =
   (maxLine * LIST_PER_LINEHEIGHT > LIST_HEIGHT)?true:false;

// Scroll 있을 경우에는 LIST_WIDTH에서 SCROLL_WIDTH을 뺀다.
if(bDrawScroll)
        list_width -= SCROLL_WIDTH;

// 실제적으로 보여야할 영영에 List 내용을 그린다.
g.setClip(LIST_X, list_y, list_width, LIST_HEIGHT);

// List가 현재 선택되 라인에 따라서 y축을 변경해야 하는 조건이 생긴다.
// 자세한 조건은 KeyPressed에서 하겠습니다.
// y축을 위로 움직여서 마지막 라인을 위로 올라오게 한다.
// 부분적으로 보이는 마지막라인의 안보이는 만큼만 위로 올린다.
if(list_yMove)
        list_y -= (maxPerLine * LIST_PER_LINEHEIGHT - LIST_HEIGHT);

for(i = firstLine, k = 0; i < lastLine; i++, k++)
{
        text_x = LIST_X;
        text_y = list_y + LIST_PER_LINEHEIGHT * k;

        if(k == (curLine - firstLine))
        {
                // 선택 바 그리기.
                g.setColor(barColor);
                g.fillRect(LIST_X, text_y, list_width, LIST_PER_LINEHEIGHT);

                // 리스트 숫자 그리기
```

```
                                g.setColor(selfontColor);
                                g.drawChar(list_num[i], text_x + 1, text_y, g.TOP|g.LEFT);
                                text_x += LIST_ITEM_WIDTH;

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

                        // 리스트 숫자 그리기
                        g.setColor(fontColor);
                        g.drawChar(list_num[i], text_x + 1, text_y, g.TOP|g.LEFT);
                        text_x += LIST_ITEM_WIDTH;

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

                // 화면에 그리는 영역을 리스트 영역으로 복구
                g.setClip(LIST_X, LIST_Y, LIST_WIDTH, LIST_HEIGHT);

                // Scroll 를 그릴것인지 안닌지.
                if(bDrawScroll)
                        drawScroll(g, maxLine, curLine);
        }


/*
** ------------------------------------------------------------------------
**  Function:
**          drawScroll
**
**  Description:
**          Scroll 를 그린다.
```

```
**
**   Input:
**         g : Graphics 객체
**         maxLine : 최대 라인
**         selLine : 현재 선택된 라인.
**
**   Return value:
**
** ------------------------------------------------------------------------
*/

          public void drawScroll(Graphics g, int totLine, int selLine)
          {
                    int scroll_height = LIST_HEIGHT;
                    int scroll_x = LIST_X + LIST_WIDTH - SCROLL_WIDTH;
                    int scroll_y = LIST_Y;

                    int scrollBar_x = scroll_x;
                    int scrollBar_y = scroll_y + ((totLine <= 1)?0:
                       (scroll_height - SCROLLBAR_HEIGHT) * selLine / (totLine - 1));

                    // 스크롤 그리기
                    g.setColor(0xc1dbf4);
                    g.drawLine(scroll_x, scroll_y, scroll_x, scroll_y + scroll_height);
                    g.setColor(0xedf1f5);
                    g.fillRect(scroll_x + 1, scroll_y, SCROLL_WIDTH - 1, scroll_height);

                    // 스크롤바 그리기
                    g.setColor(0x000000);
                    g.fillRect(scrollBar_x, scrollBar_y, SCROLLBAR_WIDTH, SCROLLBAR_HEIGHT);
          }

          public int getSelect(){}
          public char[] getChars(){}
          public String getString(){}
}
```

D. public void keyPressed(int keyCode)
public void keyReleased(int keyCode)
public int getSelect()
public char[] getChars()



가장 중여한 키 이벤트 입니다. 화면을 보시면 아래 부분에 약간의 보이는 부분이 있기 때문에 해당 마지막 라인을 선택하게 되면 전체적으로 리스트가 약간 위로 올라가야 합니다. 이러한 것은 list_yMove가 처리하고 있습니다. true, false로 구분하게 되는 것입니다. 그렇다면 언제 true, false인가 하는 조건이 있죠. 시작시에는 무조건 첫번째라인이 기준이 되어서 그려지는 것이죠..하지만 일단 마지막 라인을 선택하면 기준은 마지막 라인을 기준으로 해서 그려야합니다. 그렇다고 그리는 것을 마지막 라인부터 그리는 것은 아니고..첫번째라인을 마지막라인이 안보이는 부분 만큼 약간 올려서 그리면 마지막 라인이 딱맞추어서 보이게 되겠죠.. 이것이 핵심 포인트 입니다.

위의 여태까지 한 것을 토대로 나머지 부분이 전부 구현됩니다. 하지만..제가 전 강좌에 했었던 것 처럼 구현됩니다. 약간 복잡해 보일지도 모릅니다. 원래 List까지는 그렇게 안해도 되는데….원래 컴포넌트는 상태값이란 것이 없기 때문에 그냥 짜도 됩니다. 오히려 그게 더 효율적이지요..하지만..예전의 강좌의 예제도 만들겸해서 그냥 이렇게 했습니다. 여러분들은 컴포넌트를 할때는 기존의 paint(), keyPressed, keyReleased() 방식으로 하십시오.

```
import java.io.*;
import java.util.*;
import javax.microedition.lcdui.*;

import com.xce.lcdui.*;
import com.xce.io.*;
import com.skt.m.*;

public class PinkList
```

```
{

/*
** ------------------------------------------------------------------
**        Define 선언
** ------------------------------------------------------------------
*/

        // 한줄당 라인 높이..일반적으로 폰트 크기 보다 1정도 크게 하는데.
        // 이런 세세한것은 알아서...
        public final static int LIST_PER_LINEHEIGHT    = 15;
        public final static int LIST_ITEM_WIDTH                    = 15;    // 리스트의 숫자와 글자와의 간격

/*
** ------------------------------------------------------------------
**        변수 선언
** ------------------------------------------------------------------
*/

        int LIST_X;                // List X축
        int LIST_Y;                // List Y축
        int LIST_WIDTH;            // List Width
        int LIST_HEIGHT;           // List Height


        int firstLine;             // 현재 화면에 보이는 상단 Line
        int curLine;               // 현재 선택된 Line
        int maxLine;               // 리스트 전체 라인 개수

        int maxPerLine;            // LIST_HEIGHT에 들어가 최대 줄 개수

        int list_anchor;           // 기타 옵션
        boolean list_yMove;        // 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 fontColor = 0x000000;               // 폰트 칼라
        int selfontColor = 0xffff00;  // 선택된 폰트 칼라
        int barColor = 0x000000 ;               // 선택바 칼라


        // 스크롤 관련
        public final static int SCROLL_WIDTH          = 6;      // 스크롤의 배경 폭
        public final static int SCROLLBAR_WIDTH       = 5;      // 스크롤 바의 폭
        public final static int SCROLLBAR_HEIGHT      = 29;     // 스크롤 바의 높이


        PinkCanvas pinkCanvas;

/*
** ----------------------------------------------------------------
**  Function:
**          XList
**
**  Description:
**          생성자
**
**  Input:
**
**  Return value:
**
** ----------------------------------------------------------------------------
*/

        PinkList(PinkCanvas pinkCanvas)
        {
                this.pinkCanvas = pinkCanvas;
        }
```

```
/*
** ------------------------------------------------------------------------
**  Function:
**          setBounds
**
**  Description:
**          리스트의 영역 설정
**
**  Input:
**      x : x축
**      y : y축
**      width : 폭
**      height : 넓이
**
**  Return value:
**
** ------------------------------------------------------------------------
*/

        public void setBounds(int x, int y, int width, int height)
        {
                LIST_X = x;
                LIST_Y = y;
                LIST_WIDTH = width;
                LIST_HEIGHT = height;

                // Height에 들어갈 최대 줄개수를 구합니다.
                // 뒤부분은 화면에 조금 보이더라도 줄로 인정하는 것입니다.
                // (이부분이 줄개수의 핵심이죠)
                maxPerLine = LIST_HEIGHT / LIST_PER_LINEHEIGHT +
                  ((LIST_HEIGHT % LIST_PER_LINEHEIGHT > 0)?1:0);
        }
```

```
/*
** ------------------------------------------------------------------------
**  Function:
**          setChars
**
**  Description:
**          리스트 배열 문자 세팅
**
**  Input:
**      list : 리스트 배열
**      count : 배열 개수
**      anchor : 기타 옵션(사용하지 않는다 그냥 옵션은 옵션일 뿐이다.)
**
**  Return value:
**
** ------------------------------------------------------------------------
*/
        public void setChars(char[][] list, int count, int anchor)
        {
                int length = 0;
                int i = 0;

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

                // list_text 초기화
                // 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.
                if(list_text != null)
                {
                        for(i = 0; i < maxLine; i++)
```

```
                                        list_text[i] = null;

                        list_text = null;
                }

                // 변수 초기화
                firstLine = 0;
                curLine = 0;
                maxLine = count;
                list_anchor = anchor;

                list_text = new char[maxLine][];

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

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


        public void setString(String[] list, int count, int anchor)
        {
                int i = 0;

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

                if(list_text != null)
                {
                        for(i = 0; i < maxLine; i++)
```

```
                                        list_text[i] = null;

                                list_text = null;
                        }

                        // 변수 초기화
                        firstLine = 0;
                        curLine = 0;
                        maxLine = count;
                        list_anchor = anchor;

                        list_text = new char[maxLine][];

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


/*
** ------------------------------------------------------------------------
**  Function:
**          pinkList_EventProc
**
**  Description:
**          이벤트 프로시져.
**
**  Input:
**      status : 현재 이벤트
**      g : Graphics 객체
**      keyCode : 키값.
**
**  Return value:
**
** ------------------------------------------------------------------------
*/
```

```
	public void pinkList_EventProc(int status, Graphics g, int keyCode)
	{
		switch(status)
		{
			case PinkCanvas.EVT_PAINT:
				// 리스트 배경 색칠..
				g.setColor(0xffffff);
				g.fillRect(LIST_X, LIST_Y, LIST_WIDTH, LIST_HEIGHT);

				drawList(g);
				break;

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

							// 화면의 첫번째 라인 설정
							// LIST_HEIGHT 보다 큰 경우.
					// 마지막 라인에서 화면에 보여주는 만큼을 뺀다.
							firstLine = (maxLine * LIST_PER_LINEHEIGHT <= LIST_HEIGHT)?0:maxLine - maxPerLine;
						} else
						{
						// 첫번째 라인을 하나씩 위로 올린다.
							if(firstLine > curLine - 1)
								firstLine--;

							// 현재 라인도 위로 올린다.
							curLine--;
```

```
                                        }

                                        // 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((curLine + 1) * LIST_PER_LINEHEIGHT >
LIST_HEIGHT)
                                                list_yMove = 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(list_yMove)
                                                list_yMove     =     curLine     ==
firstLine?false:true;

                                        pinkCanvas.setRefreshArea(LIST_X, LIST_Y,
LIST_WIDTH, LIST_HEIGHT);
                                        pinkCanvas.repaintArea();
                                        break;

                                case Canvas.KEY_DOWN:
                        // 현재 라인이 마지막 라인이면 첫번째라인으로 이동한다.
                                        if(curLine >= maxLine - 1)
                                        {
                                                firstLine = curLine = 0;
                                        }else
                                        {
                                                // 첫번째 라인을 증가시킨다.
                                                if(firstLine + maxPerLine - 1 <=
curLine)
                                                        firstLine++;

                                                // 현재 라인을 증가시킨다.
```

```
                                                curLine++;
                                            }

                                            // 위와 상동이다.
                                            if((curLine + 1) * LIST_PER_LINEHEIGHT >
LIST_HEIGHT)
                                                list_yMove = true;

                                            // 위와 상동이다.
                                            if(list_yMove)
                                                list_yMove = curLine ==
firstLine?false:true;

                                            pinkCanvas.setRefreshArea(LIST_X, LIST_Y,
LIST_WIDTH, LIST_HEIGHT);
                                            pinkCanvas.repaintArea();
                                            break;
                                    }
                                    break;

                            case PinkCanvas.EVT_KEY_RELEASED:
                                    break;
                    }
            }


/*
** ---------------------------------------------------------------------------
**  Function:
**          drawList
**
**  Description:
**          List 를 그린다.
**
**  Input:
**      g : Graphics 객체
```

```
**
**   Return value:
**
** ------------------------------------------------------------------------
*/
        public void drawList(Graphics g)
        {
                // List의 마지막 라인을 찾는다.
                // 기본적으로 전체 라인 개수가 maxPerLine 작거나 같으면 마지막 라인이 전체라
인이 되고
                // 그렇지 않으면 첫번째라인에서 maxPerLine 만큼 더해서 마지막 라인을 계산한
다.
                int lastLine = (maxLine <= maxPerLine)?maxLine:
                        ((firstLine + maxPerLine   > maxLine)?maxLine:firstLine + maxPerLine);

                int i, k;

                int list_y  = LIST_Y;           // Y축이 변할 수 있다.
                int list_width = LIST_WIDTH; // 폭이 변할 수 있다.

                int text_x;           // 글자를 그리는 X축
                int text_y;           // 글자를 그리는 Y축

                // Scroll 를 그릴것인지 안닌지.
                boolean bDrawScroll =
                     (maxLine * LIST_PER_LINEHEIGHT > LIST_HEIGHT)?true:false;

                // Scroll 있을 경우에는 LIST_WIDTH에서 SCROLL_WIDTH을 뺀다.
                if(bDrawScroll)
                        list_width -= SCROLL_WIDTH;

                // 실제적으로 보여야할 영영에 List 내용을 그린다.
                g.setClip(LIST_X, list_y, list_width, LIST_HEIGHT);

                // List가 현재 선택되 라인에 따라서 y축을 변경해야 하는 조건이 생긴다.
                // 자세한 조건은 KeyPressed에서 하겠습니다.
```

```
                // y축을 위로 움직여서 마지막 라인을 위로 올라오게 한다.
                // 부분적으로 보이는 마지막라인의 안보이는 만큼만 위로 올린다.
                if(list_yMove)
                        list_y -= (maxPerLine * LIST_PER_LINEHEIGHT - LIST_HEIGHT);

                for(i = firstLine, k = 0; i < lastLine; i++, k++)
                {
                        text_x = LIST_X;
                        text_y = list_y + LIST_PER_LINEHEIGHT * k;

                        if(k == (curLine - firstLine))
                        {
                                // 선택 바 그리기.
                                g.setColor(barColor);
                                g.fillRect(LIST_X, text_y, list_width, LIST_PER_LINEHEIGHT);

                                // 리스트 숫자 그리기
                                g.setColor(selfontColor);
                                g.drawChar(list_num[i], text_x + 1, text_y, g.TOP|g.LEFT);
                                text_x += LIST_ITEM_WIDTH;

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

                        // 리스트 숫자 그리기
                        g.setColor(fontColor);
                        g.drawChar(list_num[i], text_x + 1, text_y, g.TOP|g.LEFT);
                        text_x += LIST_ITEM_WIDTH;

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

```
                // 화면에 그리는 영역을 전체 영역으로 복구
                g.setClip(LIST_X, LIST_Y, LIST_WIDTH, LIST_HEIGHT);

                // Scroll 를 그릴것인지 안닌지.
                if(bDrawScroll)
                        drawScroll(g, maxLine, curLine);
        }


/*
** ------------------------------------------------------------------------
**  Function:
**          drawScroll
**
**  Description:
**          Scroll 를 그린다.
**
**  Input:
**      g : Graphics 객체
**      maxLine : 최대 라인
**      selLine : 현재 선택된 라인.
**
**  Return value:
**
** ------------------------------------------------------------------------
*/

        public void drawScroll(Graphics g, int totLine, int selLine)
        {
                int scroll_height = LIST_HEIGHT;
                int scroll_x = LIST_X + LIST_WIDTH - SCROLL_WIDTH;
                int scroll_y = LIST_Y;

                int scrollBar_x = scroll_x;
```

```
                // 스크롤 바의 Y축 위치 계산
                int scrollBar_y = scroll_y + ((totLine <= 1)?0:(scroll_height - SCROLLBAR_HEIGHT - 2) * selLine / (totLine - 1));

                // 스크롤 그리기
                g.setColor(0xff0000);
                g.fillRect(scroll_x + 1, scroll_y + 1, SCROLL_WIDTH - 1, scroll_height);
                g.setColor(0xc166f4);
                g.drawRect(scroll_x, scroll_y, SCROLL_WIDTH, scroll_height - 1);

                // 스크롤바 그리기
                g.setColor(0x00ffff);
                g.fillRect(scrollBar_x + 1, scrollBar_y + 1, SCROLLBAR_WIDTH, SCROLLBAR_HEIGHT);
        }


/*
** ----------------------------------------------------------------------------
**  Function:
**      getSelect, getChars, getString
**
**  Description:
**          현재 선택된 것에 관련하여 리턴한다.
**
**  Input:
**
**  Return value:
**
** ----------------------------------------------------------------------------
*/

        public int getSelect()
        {
                return curLine;
        }
```

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

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

		return text;
	}

	public String getString()
	{
		return (new String(list_text[curLine]));
	}
}
```

현재까지 말로만 했던...리스트가 완성되었다...하지만 저것을 어떻게 Canvas와 연결시킬까..
간단히 MIDlet, Canvas를 준비했다.

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import java.io.*;

import com.xce.lcdui.*;
import com.skt.m.*;

public class PinkMIDlet extends MIDlet
{


/*
** ----------------------------------------------------------------
**	변수 선언
** ----------------------------------------------------------------
*/
```

```
        Display display;                        // Display 변수
        PinkCanvas pinkCanvas;          // Canvas 변수


/*
** ----------------------------------------------------------------------
**  Function:
**          startApp
**
**  Description:
**          어플리케이션이 시작되면 실행됨
**
**  Input:
**
**  Return value:
**
** ----------------------------------------------------------------------
*/

        public void startApp()
        {
                display = Display.getDisplay(this);
                pinkCanvas = new PinkCanvas(this);
                display.setCurrent(pinkCanvas);
        }


/*
** ----------------------------------------------------------------------
**  Function:
**          pauseApp
**
**  Description:
**          어플리케이션이 멈출경우 호출됨(잘 동작하지 않음)
**
```

```
**   Input:
**
**   Return value:
**
** ---------------------------------------------------------------------
*/
        public void pauseApp() {}

/*
** ---------------------------------------------------------------------
**   Function:
**            resumeApp
**
**   Description:
**            어플리케이션 pauseApp() 후에 복귀 할때 호출됨(실제로 동작하지 않음)
**
**   Input:
**
**   Return value:
**
** ---------------------------------------------------------------------
*/

        public void resumeApp(){}

/*
** ---------------------------------------------------------------------
**   Function:
**            destroyApp
**
**   Description:
**            어플리케이션 종료 될때 호출됨
**
**   Input:
**
**   Return value:
```

```
**
** ----------------------------------------------------------------
*/
        public void destroyApp(boolean con){}

}
```

```
import java.io.*;
import java.util.*;
import javax.microedition.lcdui.*;

import com.xce.lcdui.*;
import com.xce.io.*;
import com.skt.m.*;

public class PinkCanvas extends Canvas
{

/*
** ----------------------------------------------------------------
**      Define 정의
** ----------------------------------------------------------------
*/

        // 단말 LCD 정보
        public final static int LCD_WIDTH           = 120;
        public final static int LCD_HEIGHT          = 144;
        public int LCD_X;
        public int LCD_Y;



        // 이벤트 프로시져 상태
        public final static int EVT_PAINT                       = 1;     // Paint Event
        public final static int EVT_KEY_PRESSED   = 2;     // Key Pressed Event
        public final static int EVT_KEY_RELEASED = 3;     // Key Released Event
```

```
/*
** ------------------------------------------------------------------
**      변수 선언
** ------------------------------------------------------------------
*/

        private int REAREA_X;                               // 리플레쉬 영역 X
        private int REAREA_Y;                               // 리플레쉬 영역 Y
        private int REAREA_WIDTH;                   // 리플레쉬 영역 WIDTH
        private int REAREA_HEIGHT;                  // 리플레쉬 영역 HEIGHT

        private final char[] subject = {'핑', '크', '리', '스', '트'};


        PinkMIDlet      pinkMIDlet;
        PinkList        pinkList
/*
** ------------------------------------------------------------------
**  Function:
**          CyWorldCanvas
**
**  Description:
**          생성자.
**
**  Input:
**
**  Return value:
**
** ------------------------------------------------------------------
*/

        PinkCanvas(PinkMIDlet pinkMIDlet)
        {
                this.pinkMIDlet = pinkMIDlet;
```

```
                LCD_X = (this.getWidth() - LCD_WIDTH) / 2;
                LCD_Y = (this.getHeight() + 16 - LCD_HEIGHT) / 2;

                pinkList = new PinkList(this);

                char[][] exChars = {{'핑' ,'크', '리', '스', '트', '라', '인', '0', '1'},
                                    {'핑' ,'크', '리', '스', '트', '라', '인', '0', '2'},
                                    {'핑' ,'크', '리', '스', '트', '라', '인', '0', '3'},
                                    {'핑' ,'크', '리', '스', '트', '라', '인', '0', '4'},
                                    {'핑' ,'크', '리', '스', '트', '라', '인', '0', '5'},
                                    {'핑' ,'크', '리', '스', '트', '라', '인', '0', '6'},
                                    {'핑' ,'크', '리', '스', '트', '라', '인', '0', '7'},
                                    {'핑' ,'크', '리', '스', '트', '라', '인', '0', '8'},
                                    {'핑' ,'크', '리', '스', '트', '라', '인', '0', '9'}};

                String[] exString = {"핑크리스트01",
                                                    "핑크리스트02",
                                                    "핑크리스트03",
                                                    "핑크리스트04",
                                                    "핑크리스트05",
                                                    "핑크리스트06",
                                                    "핑크리스트07",
                                                    "핑크리스트08",
                                                    "핑크리스트09"};

                pinkList.setChars(exChars, exChars.length, -1);
                pinkList.setBounds(2, 16, this.getWidth() - 4, this.getHeight() - 2);

        }



/*
** ---------------------------------------------------------------
**   Function:
**            paint
```

```
**
**   Description:
**
**
**   Input:
**
**   Return value:
**
** ------------------------------------------------------------------
*/

        public void paint(Graphics g)
        {
                main_EventProc(EVT_PAINT, g, -1);
        }


/*
** ------------------------------------------------------------------
**   Function:
**           setRefreshArea
**
**   Description:
**           repaint 영역을 정한다.
**
**   Input:
**                x :
**                y :
**                width :
**                height :
**
**   Return value:
**
** ------------------------------------------------------------------
*/
        public void setRefreshArea()
```

```
        {
                setRefreshArea(LCD_X, LCD_Y, LCD_WIDTH, LCD_HEIGHT);
        }

        public void setRefreshArea(int x, int y, int width, int height)
        {
                REAREA_X = x;
                REAREA_Y = y;
                REAREA_WIDTH = width;
                REAREA_HEIGHT = height;
        }


        public void repaintArea()
        {
                repaint(REAREA_X, REAREA_Y, REAREA_WIDTH, REAREA_HEIGHT);
                serviceRepaints();
        }

/*
** ----------------------------------------------------------------------
**  Function:
**          keyPressed
**
**  Description:
**          -  키 누름 처리
**
**  Input:
**              // SK-VM Specific
**              KEY_CLR                 = 8;
**
**          KEY_POUND           = 35;
**              KEY_NUM0        = 48;
**          KEY_NUM1   = 49;
**          KEY_NUM2   = 50;
**          KEY_NUM3   = 51;
```

```
**          KEY_NUM4  = 52;
**          KEY_NUM5  = 53;
**          KEY_NUM6  = 54;
**          KEY_NUM7  = 55;
**          KEY_NUM8  = 56;
**          KEY_NUM9  = 57;
**          KEY_STAR  = 42;
**
**              KEY_COML      = 129;
**              KEY_COMC      = 130;  // RESERVED
**              KEY_COMR      = 131;
**
**              KEY_UP        = 141;
**              KEY_LEFT      = 142;
**              KEY_RIGHT     = 145;
**              KEY_DOWN      = 146;
**              KEY_FIRE      = 148;
**
**              KEY_CALL      = 190;
**              KEY_END               = 191;
**
**              KEY_FLIP_OPEN = 192;
**              KEY_FLIP_CLOSE        = 193;
**              KEY_VOL_UP            = 194;
**              KEY_VOL_DOWN= 195;
**
**  Return value:
**
** ------------------------------------------------------------------
*/


        public void keyPressed(int keyCode)
        {
                main_EventProc(EVT_KEY_PRESSED, null, keyCode);
        }
```

```
/*
** ---------------------------------------------------------------------
**   Function:
**            keyReleased
**
**   Description:
**
**
**   Input:
**
**   Return value:
**
** ---------------------------------------------------------------------
*/

        public void keyReleased(int keyCode)
        {
                main_EventProc(EVT_KEY_RELEASED, null, keyCode);
        }


/*
** ---------------------------------------------------------------------
**   Function:
**            main_EventProc
**
**   Description:
**            - 메인 프로시져.
**
**   Input:
**
**   Return value:
**
** ---------------------------------------------------------------------
*/
```

```
	public void main_EventProc(int status, Graphics g, int keyCode)
	{
		switch(status)
		{
			case EVT_PAINT:
				g.setColor(0x00ff00);
				g.fillRect(LCD_X, LCD_Y, LCD_WIDTH, LCD_HEIGHT);

				g.setColor(0xff6daa);
				g.fillRect(LCD_X, LCD_Y, LCD_WIDTH, 16);
				g.setColor(0xff0000);
				g.drawRect(LCD_X, LCD_Y, LCD_WIDTH - 1, 15);

				g.setColor(0x000000);
				g.drawChars(subject, 0, subject.length, LCD_X +
LCD_WIDTH / 2, LCD_Y + 1, g.HCENTER | g.TOP);
				break;

			case EVT_KEY_PRESSED:
				break;

			case EVT_KEY_RELEASED:
				break;
		}

		pinkList.pinkList_EventProc(status, g, keyCode);
	}
}
```

설마 나보고 MIDlet, Canvas가 이해가 안가다고 말씀하시면..큰일입니다. 더 공부하십시오. 위의 MIDlet, Canvas는 아주 기본적인 것으로써 이해를 못하면..자기의 능력이 부족하다고 생각합니다. 저를 욕해도 좋습니다. 그러나 부족한 것은 인정해야 합니다. 더 공부 하십시오.

일단은 이렇게 해서 에뮬레이터에서 돌아가는 리스트를 만들었습니다. 처음에 말씀드렸던 더 발전적인 리스트라고 할수 있습니다. 더 자연스럽고 먼가 신경쓰지 않아도 되는..

애개 하시는 분들도 있으실겁니다. 겨우 이 정도의 리스트 가지고..그렇게 난리 부르스를 쳤다 말인가.. 하지만 이것이 전부가 아닙니다. 다음 강좌에서는 이 리스트를 한단계 더 업그레이드 시키겠습니다. ㅎㅎㅎ 여기까지 보고 놀라신분은 솥뚜껑 보고 놀라시고..별거다 하시는 분은 다음 강좌에 더 파워업된 기능을 가진 리스트를 보고도 안놀라실겁니다. ㅜㅜ

다음 강좌는 약간의 힌트를 드리자면..기존의 리스트는 위와 같이 정형화 되면서 해당 것으로도 모든 기능이 가능했습니다. 하지만..UI가 이뻐지고 요구하는 것이 많아지자 위의 리스트만 가지고는 부족하다고 생각했습니다. 그래서 전 어떻게 하면 더 좋게 리스트를 꾸밀수 있을까..에 대한 답변으로 마진(Margin)이라는 것을 생각했습니다. 일명 여백이죠 ㅜㅜ. 보통 리스트를 만들 때 마진(Margin)을 생각하지는 않습니다. 하지만 더욱더 낳은 UI를 제공하기 위해서는 마진이라는 것이 필수다라는 것을 깨닫게 되었습니다. 한차원 더 편리해지는 리스트이지요..^^ 그래서 다음 강좌에서는 위의 리스트에 마진을 넣어서 보여드리겠습니다. 마진의 기능은 또한 여러분들은 감동시킬겁니다.

그럼 즐거운 하루 되십시오. 제가 요즘에 쩜 한가해지는 줄 알았는데 사정이 여의치가 않네요.. 그래서 자주 글을 쓸시간이 없네요.. 그래도 계속적으로 쓰다가 보면 강좌 한편이 만들어 지면..기쁘답니다. 또한 제 강좌를 보면서 공부하시는 여러 학생들을 혹은 직장인 분들이 메일과 힘을 주실때마나 감동의 눈물이 흘른답니다. 여태까지 얼마 안됩니다. 10손가락도 안되지요..ㅎㅎㅎ 비참한 현실입니다.ㅜㅜ
내가 왜 이렇게 강좌를 써야 하는지..ㅜㅜ

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.