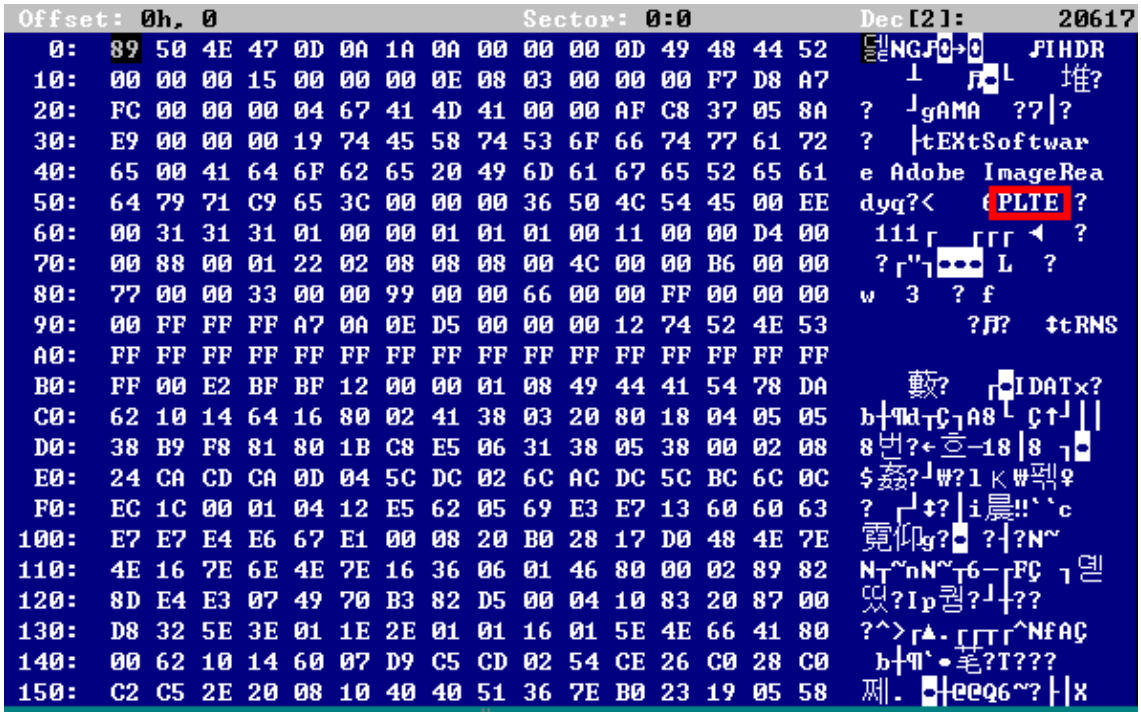
게임..ㅋㅋㅋ 내손안에 있소이다...

안녕하세요 핑크레드 입니다. 봄이 왔는데..날은 좋지 않군요.. 저번에 꽃나들이 갔었는데.. 무지 좋더군요..역시 봄입니다. ㅋㅋㅋ 사무실에도 봄이 왔는지..가물가물 하네요.

많은 분들이 제 강좌를 봐주시고 계셔서 영광입니다. 도움이 될지는 모르겠지만..정보 공유라는 것이 상당히 중요하다고 보았습니다. 솔직히 기술력이라고 하면 프로그래머는 자기만의 무언가로써 남에게 공유하는 것을 싫어합니다. 그 기술을 남에게 줘버리면 자기는 아무것도 없다고 생각하기 때문이죠. 사실은 그말이 맞기는 맞는데...사회생활을 해보니 그렇다고 내가 그 기술을 가지고 있으면 언제나 제가 그일을 해야하는 것이고 다음 사람한테 되물림 같은게 되지 않는다는 크나큰 단점이 있습니다. 언제까지 프로그래만 짜고 있기도 그렇죠... 그런 생각에 처음부터 시작했었고 .. 제 강좌를 보시고 도움이 되시는 분들을 볼때마다 가슴이 뿌듯하면서 인터넷이라는 것이 정말 대단하구나라는 것도 세삼 느낍니다. ㅋㅋㅋ 오늘은 웬지 감정적이군요. 날씨가 않좋아서 그런가 봅니다. 이런날은 또 잔소리 하는게 제 맛입니다. ㅋㅋㅋ 일단..쉼호흡을 크게 쉬고 멀찌감치 뒤에서 나의 일들을 살펴봅니다. 무엇을 하고 있는지..매일 앉아서 코딩하는 것 말고..놀고..먹고.. 등등…

오늘은 새로운 기술을 알려드리겠습니다. 이미지 기술인데..아시는 분들은 다 아시리라 생각되지만 참으로 좋은 기술이라고 생각됩니다. PNG 이미지 파일을 이용하여 색상 바꾸기 입니다. ㅋㅋㅋ 다 아신다구요..저도 이 기술을 안지는 몇 달 안되지만 꽤 쓸만합니다. PNG 파일을 저장할시에 포토샵에서 Ctrl + Shift + Alt + S 를 눌러서 저장하시면 해당 이미지의 칼라개수가 나옵니다. 그래서 그 정보 그대로 저장을 하시면 PNG 이미지가 무척이나 작아집니다. 일단 PNG 작게 하는 팁입니다. 꼭 저장시에 PNG8 로 저장하셔야 합니다. 그리고 나서 파일을 열어보면 아래와 같은 코드들을 보면 빨간 네모 부분이 팔레트가 시작되는



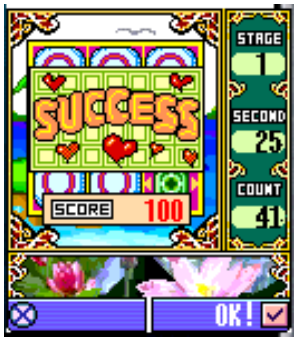
부분입니다. 그러므로 그 다음부터 한바이트씩 RGB 가 되는 것이지요 3 바이트가 하나의

색상을 정하는 것이지요. 그럼 이 팔레트의 색상을 바꾸어 버리면 기존의 이미지에도 똑같이 색상이 변화됩니다. 그래서 총 색상 개수대로 RGB 를 읽어서 바꾸신후에 이용하시면 됩니다. 응용은 알아서 하시고..ㅋㅋㅋ 이로 인해서 얻는 이득은 직접 생각해 보십시오.

아주 좋은 정보를 드렸습니다. ^^ 가끔씩은 느끼는 부분이 있지만 강의를 하다가 좋은 정보나만 하는 정보를 주시면 가끔씩 강좌에 올리도록 하겠습니다. 몰론 정보를 주신 분의 명의로 해드리지요.. 여태까지 아무도 없었습니다. 우리나라 국민성이 원래 나서는거 무지 싫어하죠.. 공유는 돈되는 것만하고 ㅡㅡ; ㅋㅋㅋ 여튼 서로 도와가면서 사는 것이지요. 근데 CP 끼리는 이런 부분들이 노하우라서 공유를 전혀 하지 않을 것 같습니다. ㅋㅋㅋ 난 쫓겨날지도 몰라요..

오늘은 이만 하고 날씨도 꿀꿀하고 해서 강좌 2 편을 한번에 다 올립니다. 차근 차근 보시면서 하면은 좋을 듯 합니다. 한번에 2 개가 올라오면 무지 힘드시겠지만 열심히 하십시오. 그럼 갑니다.

8 번째는 성공시에 보여주는 화면 입니다.



CSound.java

```
package Cube;

import com.skt.m.*;
import java.io.*;
import java.util.*;

public class CSound
{
	public final static int SOUND_NUM   = 1;	// 사운드 파일의 개수.
	public final static int STOP        = 10;

	AudioClip clip;
	byte buffer[][] = new byte[SOUND_NUM][];	// 사운드 파일이 많아질 경우를 대비해 배열로 잡음.

	//사운드 로딩 및 생성
	public CSound()
	{
		try {
			clip = AudioSystem.getAudioClip("mmf");
			for (int i=0; i<SOUND_NUM; i++) {
				InputStream is = getClass().getResourceAsStream("/Cube/snd/sound" + i + ".mmf");
```

```
                                buffer[i] = new byte[is.available()];
                                is.read(buffer[i]);
                        }
                }catch (Exception ex) {
                }
        }

        // 사운드 플레이
        // playSound(번호) 형식으로 호출하면 해당번호 사운드 재생
        // playSound(STOP) - 사운드 출력정지
        public synchronized void playSound(int num) {
                try {
                        if(num != STOP){
                                clip.open(buffer[num], 0, buffer[num].length);
                                clip.play();
                        }else{
                                clip.stop();
                        }
                }catch (Exception ex) {
                }finally {
                        try     {
                                clip.close();
                        }catch(Exception ex2) {
                        }
                }
        }
}
```

CubeEngine.java

```
package Cube;

import javax.microedition.lcdui.*;
import java.util.*;
import java.io.*;
import com.skt.m.*;


public class CubeEngine
{
        // mode 부분 상수들.
        public final static int LOGO                    = 0;
        public final static int TITLE                   = 1;
        public final static int LEVEL                   = 2;
        public final static int MISSION                         = 3;
        public final static int GAME                    = 4;
        public final static int RUNMENU                         = 5;
        public final static int SUCCESS                         = 6;
        public final static int CLEAR                           = 7;

        // 키입력부분 상수들.
        public final static int KEY_UP                          = 141;
        public final static int KEY_LEFT                = 142;
        public final static int KEY_RIGHT               = 145;
        public final static int KEY_DOWN                = 146;
        public final static int SOFT_LEFT               = 129;
        public final static int SOFT_RIGHT              = 131;
```

```
	// 멤버 변수
	public int 			cellsize;
	public int 			xySize;
	public int 			cx, cy;
	// 커서의 x, y 값
	public int 			menuSwitch = 1;			// 메뉴
선택을 위해서 쓰이는 모드변수
	public int 			levelSwitch = 1;
	public int 			runSwitch = 1;
	private int 			stageNumber;			// 현재
스테이지의 넘버
	private int 			mode;
	// 화면에 따른 키이벤트를 구분하기 위해서 쓰이는 모드변수.
	private int 			score;
	static int 		swap;					// cell 의
이동시 값 치환을 위한 임시변수
	static int[][] puzzle_num = new int[4][4];  // 퍼즐의 각 픽셀(사각형)에 들어갈
숫자배열
	static int[][] show_puzzle = new int[4][4];  // 미션제시 퍼즐 숫자배열
	String[] stage = new String[3];				//모든 Stage 정
보를 담는 배열


	// 참조클래스 선언
	private CubeCube cube;
	private CubeCanvas	canvas;				//
Canvas
	private CSound	sound;


	// Constructor
	public CubeEngine(CubeCanvas canvas)
	{
		this.canvas = canvas;
		sound = new CSound();

		stage[0] = new String("111112110");
	// 테스트용
		stage[1] = new String("111112110");
		stage[2] = new String("111112110");
/*
		stage[0] = new String("111212110");
	// 포팅용
		stage[1] = new String("111324110");
		stage[2] = new String("131424130");
*/
	}

	// GET & SET method
	public int getMode()
	{
		return mode;
	}

	public void setMode(int mode)		// 클리어 때 빙고에서 이걸 바꿈..
	{
		this.mode = mode;
```

```
	}

	public int getStageNumber()
	{
		return stageNumber;
	}

	public void setStageNumber(int num)
	{
		stageNumber = num;
	}

	public int[][] getShowPuzzle()
	{
		return show_puzzle;
	}

	public int[][] getPuzzleNum()
	{
		return puzzle_num;
	}

	public void setCube(CubeCube cube)
	{
		this.cube = cube;
	}


	public void setPuzzle(int stageNum)
	{
		xySize = 3;
		cellsize=24;

		switch (stageNum)
		{
			case 1:
				setShowNum(3,0);
				setPuzzleNum(3,0);	//  사이즈, 레벨
			break;
			case 2:
				setShowNum(3,1);
				setPuzzleNum(3,1);
			break;
			case 3:
				setShowNum(3,2);
				setPuzzleNum(3,2);
			break;
		}
	}

	public void setShowNum(int size, int level)
	{
		int index=0;
		for(int i=0; i<size; i++){
			for(int j=0; j<size; j++){
				show_puzzle[i][j]                                    =
Integer.parseInt(stage[level].substring(index++,index));
			}
```

```
                }
        }


        public void setPuzzleNum(int size, int level)
        {
                int random = 0;
                random = getRandomInt(8);           // 0-7 랜덤출력
                int ran = 1;
                for(int i=0; i<size; i++){
                        for(int j=0; j<size; j++){
                                if (random > size*size-1) random = 0;
                                ran                                                     =
Integer.parseInt(stage[level].substring(random++,random));
                                if (ran==0) {cx=i; cy=j;}                       //  커서
위치 초기화 하기..
                                puzzle_num[i][j] = ran;
                        }
                }
        }

        static int getRandomInt(int limit) {          // 랜덤하게 숫자 뽑기. limit 이내.
                Random rnd = new Random();
                int number = rnd.nextInt();
                number = (number >>> 16) & 0xffff;
                number /= (0xffff/ limit);
                return number;
        }

        public void showPuzzle()
        {
                mode = MISSION;
                canvas.drawShowPuzzle(stageNumber);
        }


        public void gameStart()
        {
                System.gc();
                canvas.paintCell();
        }

        // 퍼즐셀의 미션완료 검사 (성공하면 true 리턴)
        public boolean isBingo()
        {
                int nCount = 0;
                for (int j=0; j < canvas.sizeMAX; j++){
                        for (int i=0; i < canvas.sizeMAX; i++){
                                if ( puzzle_num[i][j] != show_puzzle[j][i]){
                                        return false;
                                }
                        }
                }
                return true;
        }

        // 가중치를 부여한 점수계산
        public int calculateScore()
```

```
	{
		int valuePoint = 0;
		switch(stageNumber)
		{
			case 1:
				valuePoint = 25;
			break;
			case 2:
				valuePoint = 12;
			break;
			case 3:
				valuePoint = 10;
			break;
		}
		score = (800 - canvas.times -(canvas.moveCount*valuePoint))*10;
		// 점수계산.
		if(score < 100){score = 100;}			// 최하 100 점.
		return score;
	}



	public void keyPress(int key)
	{
		switch (mode)
		{
			case LOGO:					// Logo
			case TITLE:
				keyPressInTitle(key);
			break;

			case LEVEL:					// Show Level
				keyPressInLevel(key);
			break;

			case MISSION:				// ShowPuzzle
				keyPressInShowPuzzle(key);
			break;

			case GAME:					// In Game
				keyPressInGame(key);
			break;

			case RUNMENU:				// Game Menu
				keyPressInRunMenu(key);
			break;

			case SUCCESS:				// Stage Clear
				keyPressInClear(key);
			break;

			case CLEAR:					// Show Clear
Picture
				keyPressInClearPicture(key);
			break;

		}
	}
```

```
	public void keyPressInTitle(int key)
	{
		if(mode == LOGO)
		{
			canvas.showMainMenu();
			mode = TITLE;
			return;
		}
		if ((menuSwitch != 1) && (key == KEY_UP))		// 위방향
		{
			menuSwitch--;
			canvas.showMainMenu();
			sound.playSound(0);
			// 사운드파일 번호
		}
		else if ((menuSwitch != 5) && (key == KEY_DOWN))  // 아래방향
		{
			menuSwitch++;
			canvas.showMainMenu();
			sound.playSound(0);
			// 사운드파일 번호
		}
	}


	public void selectMenu()
	{
		switch (menuSwitch)			// 현재의 선택레벨을 가지고 감.
		{
			case 1:
				canvas.drawLevelMenu();		// 레벨메뉴 보이
기.
				mode = LEVEL;
			break;

			case 2:
				// 게임설명 (Instruction)
				// 이부분은 레벨메뉴를 띄우는 것과 동일한 방식으로
작성 됨
			break;

			case 3:
				// 저작권 정보 (Credits)
				// 이부분은 레벨메뉴를 띄우는 것과 동일한 방식으로
작성 됨
			break;

			case 4:
				// 환경설정 (Settings)
				// 이부분은 레벨메뉴를 띄우는 것과 동일한 방식으로
작성 됨
			break;

			case 5:
				cube.gameEnd();		// 게임종료 (Exit)
```

```
                        break;
                }
        }



        public void keyPressInLevel(int key)
        {
                if ((levelSwitch != 1) && (key == KEY_UP))                      //  위방
향
                {
                        levelSwitch--;
                        canvas.drawLevelMenu();
                        sound.playSound(0);
                                // 사운드 출력
                }
                else if ((levelSwitch != 3) && (key == KEY_DOWN))   // 아래방향
                {
                        levelSwitch++;
                        canvas.drawLevelMenu();
                        sound.playSound(0);
                }
        }


        public void selectLev()
        {
                canvas.times = 0;
                switch (levelSwitch)                            //  현재의  선택레벨을  가
지고 감.
                {
                        case 1:                                                 //
Level 1 선택.
                                setStageNumber(1);
                                setPuzzle(1);
                                showPuzzle();
                        break;

                        case 2:
                                // 추가형식.
                        break;

                        case 3:
                                // 추가형식.
                        break;
                }
        }


        public void keyPressInShowPuzzle(int key)
        {
                if (key == SOFT_LEFT)                   // 왼쪽소프트
                {
                        mode = GAME;
                        gameStart();
                }
        }
```

```
	public void keyPressInGame(int key)
	{
		if(key == SOFT_LEFT){
			canvas.timego = false;
			showPuzzle();				//  게임중 쇼퍼즐 보여주기
		}else{
			switch (key)
			{
				case KEY_UP:			// 위방향키
					if (cy > 0) {
						swap = puzzle_num[cx][cy];
						puzzle_num[cx][cy] = puzzle_num[cx][cy-1];
						cy--;
						puzzle_num[cx][cy] = swap;
						canvas.drawMove(cx, cy, cx, cy+1, cellsize, puzzle_num[cx][cy+1]);
					}
				break;

				case KEY_DOWN:			// 아래
					if (cy < xySize-1) {
						swap = puzzle_num[cx][cy];
						puzzle_num[cx][cy] = puzzle_num[cx][cy+1];
						cy++;
						puzzle_num[cx][cy] = swap;
						canvas.drawMove(cx, cy, cx, cy-1, cellsize, puzzle_num[cx][cy-1]);
					}
				break;

				case KEY_LEFT:			// 왼쪽
					if (cx > 0) {
						swap = puzzle_num[cx][cy];
						puzzle_num[cx][cy] = puzzle_num[cx-1][cy];
						cx--;
						puzzle_num[cx][cy] = swap;
						canvas.drawMove(cx, cy, cx+1, cy, cellsize, puzzle_num[cx+1][cy]);
					}
				break;

				case KEY_RIGHT:			// 오른쪽
					if (cx < xySize-1) {
						swap = puzzle_num[cx][cy];
						puzzle_num[cx][cy] = puzzle_num[cx+1][cy];
```

```
                                                cx++;
                                                        puzzle_num[cx][cy]      =
swap;
                                                        canvas.drawMove(cx,   cy,
cx-1, cy, cellsize, puzzle_num[cx-1][cy]);
                                                }
                                        break;
                                }
                        }
                }


        public void keyPressInRunMenu(int key)
        {
                if ((runSwitch != 1) && (key == KEY_UP))                // 위 방향
                {
                        runSwitch--;
                        canvas.runMenu();
                }
                else if ((runSwitch != 5) && (key == KEY_DOWN))   // 아래방향
                {
                        runSwitch++;
                        canvas.runMenu();
                }
        }

        public void keyPressInClear(int key)
        {
                if (key == SOFT_RIGHT)              // 오른쪽소프트
                {
                        mode = CLEAR;
                        canvas.drawClearPic(stageNumber);
                }
        }

        public void keyPressInClearPicture(int key)
        {
        }
}
```

CubeCube.java

```
package Cube;

import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import java.io.*;
import com.skt.m.*;

public class CubeCube extends MIDlet
{
        public CubeCanvas   canvas;
        static Graphics g;
        Display display;

        public CubeCube()
    {
                canvas = new CubeCanvas(this);
```

```
                Device.setBacklightEnabled(true);          // 키가 눌려질때 마다 백
라이트가 들어오는 기능설정 (기본적으로 true 로 설정)
                Device.enableRestoreLCD(true);             // 통화후 이전화면 복구설
정. SUSPEND 모드에서 돌아올때 화면복구를 위해
                loadLogo();
        }

        protected void startApp()
    {
                display = Display.getDisplay(this);
        canvas.active = true;
                display.setCurrent(canvas);
                BackLight.on(0);                    // 백라이트 On 으로 설정. 파라미터
0 이면 무한대 켜짐.
    }

    protected void pauseApp()
    {
        canvas.active = false;
    }

    public void resumeApp()
    {
    }

        public void destroyApp(boolean uc)
    {
                display.setCurrent(null);
                notifyDestroyed();
    }

        public void loadLogo()
        {
                try{
                        canvas.logo    =    Image.createImage("/Cube/img/logo.png");

                }catch(IOException e){
                        e.printStackTrace();
                }
        }

        public void gameEnd()
        {
                destroyApp(false);
        }
}
```

CubeCanvas.java

```
package Cube;

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


public class CubeCanvas extends Canvas implements Runnable, CommandListener
{
```

```
        // 상수선언
        public final static int SCREEN_WIDTH        = 128;
        public final static int SCREEN_LENGTH       = 143;

        // 이미지 변수선언
        static Image title, mt1, mt2, mt3, mt4, mcur, board, clear1, clear2, clear3, bt;

        static Image cell2_1, cell2_2, cell2_3, cell2_4, cursor, success, clscore, logo;

        // 참조클래스 선언
        private CubeCube          cube;
        public  CubeEngine                engine;                   // 게임의 로직부
분을 담당하는 클래스를 하나더 생성함.

        // 변수선언
        public boolean            active = true;
        public boolean            init = true;
        public boolean            timego;                           // 타이머의 진행
을 조절하는 On-Off 변수.
        int[][] puzzle_num = new int[4][4];                 // 게임중 퍼즐을 저장하는
배열
        int[][] show_puzzle = new int[4][4];        // 미션 퍼즐을 저장하는 배열
        public int                        moveCount;                        //  커서
의 움직인 횟수를 카운드하는 변수.
        public int                        sizeMAX;                  // 셀의 크기를 결
정 (3x3, 4x4)
        public int                        times;                            //  타이
머 누적시간 변수
        private int                       tc;
        // 주기적 repaint 와 타이머를 함께 쓰기위한 변수.
        private Thread            thread;
        static Graphics           g;

        // Command 버튼선언
        private Command cmd1 = new Command("START", 2, 0);
        private Command cmd2 = new Command("GO!", 4, 0);
        private Command cmd3 = new Command("MAP", 2, 0);
        private Command cmd4 = new Command("MENU", 8, 1);
        private Command cmd5 = new Command("BACK", 2, 0);
        private Command cmd6 = new Command("SELECT", 4, 1);
        private Command cmd7 = new Command("OK!", 4, 1);
        private Command cmd8 = new Command("", 3, 0);             // NULL


        // Constructor
        public CubeCanvas(CubeCube cube)
        {
                this.cube = cube;
                engine = new CubeEngine(this);
                engine.setCube(cube);
                loadImage();
                thread = new Thread(this);
                thread.start();
        }



        protected void paint(Graphics g) {
```

```
		this.g = g;
		if(init){
			g.drawImage(logo, 0, 22, 0);
			init = false;
		}
	}


	// 메인메뉴 띄우기
	public void showMainMenu(){
		drawBox(20, 25, 88, 94);
		g.setColor(255, 255, 255);
		g.drawString("Start Game", 64, 38, 16|1);
		g.drawString("Instruction", 64, 52, 16|1);
		g.drawString("Credits", 64, 66, 16|1);
		g.drawString("Settings", 64, 80, 16|1);
		g.drawString("Exit", 64, 94, 16|1);
		drawCursor();
		addCommand(cmd6);
		setCommandListener(this);
		repaint();
		serviceRepaints();
	}

	// 메인메뉴 선택커서표시
	public void drawCursor()
	{
		switch(engine.menuSwitch)
		{
			case 1:
				g.drawImage(mcur, 27, 43, 0);
				g.drawImage(mcur, 94, 43, 0);
			break;
			case 2:
				g.drawImage(mcur, 25, 43+(14*1), 0);
				g.drawImage(mcur, 97, 43+(14*1), 0);
			break;
			case 3:
				g.drawImage(mcur, 35, 43+(14*2), 0);
				g.drawImage(mcur, 86, 43+(14*2), 0);
			break;
			case 4:
				g.drawImage(mcur, 32, 43+(14*3), 0);
				g.drawImage(mcur, 89, 43+(14*3), 0);
			break;
			case 5:
				g.drawImage(mcur, 42, 43+(14*4), 0);
				g.drawImage(mcur, 78, 43+(14*4), 0);
			break;
		}
	}


	// 넘겨준 크기값으로 뒷배경사각형을 그리는 부분
	public void drawBox(int x, int y, int w, int h)
	{
		g.setColor(0, 0, 0);
		g.fillRect(x, y, w, h);
```

```
		g.setColor(0, 109, 85);
		g.fillRect(x+3, y+3, w-6, h-6);
		g.setColor(255, 219, 0);
		g.drawRect(x+1, y+1, w-3, h-3);
		g.drawImage(mt1, x, y, 0);
		g.drawImage(mt2, x+w-15, y, 0);
		g.drawImage(mt3, x, y+h-16, 0);
		g.drawImage(mt4, x+w-15, y+h-16, 0);
	}


	// 이미지 로딩
	public void loadImage()
	{
		try
	{
			mt1 = Image.createImage("/Cube/img/m01.png");
			mt2 = Image.createImage("/Cube/img/m02.png");
			mt3 = Image.createImage("/Cube/img/m03.png");
			mt4 = Image.createImage("/Cube/img/m04.png");
			mcur = Image.createImage("/Cube/img/mcur.png");
			board	= Image.createImage("/Cube/img/board.png");
			clear1	= Image.createImage("/Cube/img/clear1.png");
			clear2	= Image.createImage("/Cube/img/clear2.png");
			clear3	= Image.createImage("/Cube/img/clear3.png");
			bt		= Image.createImage("/Cube/img/bt.png");
			cell2_1	= Image.createImage("/Cube/img/cell2-1.png");
			cell2_2	= Image.createImage("/Cube/img/cell2-2.png");
			cell2_3	= Image.createImage("/Cube/img/cell2-3.png");
			cell2_4	= Image.createImage("/Cube/img/cell2-4.png");
			cursor	= Image.createImage("/Cube/img/cursor.png");
			success = Image.createImage("/Cube/img/success.png");
			clscore = Image.createImage("/Cube/img/cl-score.png");

		}catch(IOException ex){
			System.out.println("File not find");
		}
	}


	//레벨선택 메뉴 그리기
	public void drawLevelMenu()
	{
		drawBox(25, 35, 78, 72);
		g.setColor(255, 255, 255);
		g.drawString("[Level 1]", 65, 48, 16|1);
		g.drawString("[Level 2]", 65, 64, 16|1);
		g.drawString("[Level 3]", 65, 80, 16|1);
		drawCursor2();
		addCommand(cmd8);
		addCommand(cmd6);
		setCommandListener(this);
		repaint();
		serviceRepaints();
	}

	// 레벨메뉴 선택커서
	public void drawCursor2()
```

```
        {
                switch(engine.levelSwitch)
                {
                        case 1:
                                g.drawImage(mcur, 30, 53+(16*0), 0);
                                g.drawImage(mcur, 94, 53+(16*0), 0);
                        break;
                        case 2:
                                g.drawImage(mcur, 30, 53+(16*1), 0);
                                g.drawImage(mcur, 94, 53+(16*1), 0);
                        break;
                        case 3:
                                g.drawImage(mcur, 30, 53+(16*2), 0);
                                g.drawImage(mcur, 94, 53+(16*2), 0);
                        break;
                }
        }


        // 화면오른편 게이지보드
        public void drawBar()
        {
                g.setColor(252, 218, 4);
                g.drawLine(97, 0, 97, 108);
                g.drawImage(board, 98, 0, 0);
                g.setColor(220, 255, 172);
                g.fillRect(109, 20, 10, 9);
                g.setColor(0, 0, 0);
                g.drawString(""+(engine.getStageNumber()),          118,          18,
Graphics.TOP|Graphics.RIGHT);
                repaint();
        }

        // 미션제시화면의 표시
        public void drawShowPuzzle(int stageNum)
        {
                drawBar();
                                                // 오른편 점수표 그리기
                drawBackImg();
                                              // 배경이미지 그리기
                g.setColor(0, 0, 0);
                timer(times);
                drawCount();

                if(engine.getMode() != engine.GAME)
                {
                        removeCommand(cmd6);
                        removeCommand(cmd8);
                        addCommand(cmd1);
                        setCommandListener(this);
                }
                drawBox(18, 8, 93, 115);
                show_puzzle = engine.getShowPuzzle();

                sizeMAX = 3;
                for(int i=0; i < sizeMAX; i++){
                        for(int j=0; j < sizeMAX; j++){
                                drawCell(25 * i +28 , 25 * j +27, show_puzzle[j][i]);
```

```
                        }
                }

                g.setColor(255, 255, 255);
                g.drawString("YOUR MISSION", 28, 14, 0);
                g.drawString("STAGE "+stageNum, 43, 103, 0);
                repaint();
        }

        // 뒷배경 이미지 그리기
        public void drawBackImg()
        {
                g.drawImage(bt, 0, 109, 0);

                switch(engine.getStageNumber())
                {
                        case 1:
                                g.drawImage(clear1, 0, 0, 0);
                        break;

                        case 2:
                                g.drawImage(clear2, 0, 0, 0);

                        break;

                        case 3:
                                g.drawImage(clear3, 0, 0, 0);
                        break;
                }
        }

        // 이동시킨 셀 그리기
        public void drawMove(int cX, int cY, int oX, int oY, int size, int swap)
        {
                g.drawImage(cursor, cX*(size+1)+12, cY*(size+1)+17, 0);
                drawCell(oX*(size+1)+12, oY*(size+1)+17, swap);
                drawCount();
                if(cX == 2 && cY == 2)                                  //  특정
위치에 왔을때만 Bingo 검사
                {
                        if (engine.isBingo())
                        {
                                engine.setMode(engine.SUCCESS);
                                drawClear();
                        }
                }
                repaint();
        }


        // SUCCESS 메세지 출력
        public void drawClear()
        {
                removeCommand(cmd3);
                removeCommand(cmd4);
                addCommand(cmd8);               // NULL
                addCommand(cmd7);               // OK!
                setCommandListener(this);
```

```
                timego = false;
                g.drawImage(success, 10, 26, 0);

                g.setColor(0, 0, 0);
                g.fillRect(16, 83, 67, 15);
                g.setColor(252, 218, 172);
                g.fillRect(17, 84, 65, 13);

                g.setColor(255, 0, 0);
                g.drawImage(clscore, 20, 86, 0);
                g.drawString(""+engine.calculateScore(),          80,          84,
Graphics.TOP|Graphics.RIGHT);

                if(engine.getStageNumber() == 3){gameClear();}          // 3 판째
가 마지막. 메세지 보여주기
        }


        // 마지막 스테이지임을 알림
        public void gameClear()
        {
                g.setColor(0, 0, 0);
                g.fillRect(9, 63, 80, 20);
                g.setColor(255, 255, 0);
                g.fillRect(10, 64, 78, 18);
                g.setColor(0, 0, 0);
                g.drawString("Game Cleared", 16, 66, 20);
        }


        // 본게임 화면의 표시
        public void paintCell()
        {
                drawBar();
            // 왼쪽 점수표 그리기
                drawBackImg();                                                  //
배경이미지 그리기
                g.setColor(0, 0, 0);
                g.drawRect(9, 14, 79, 79);                              // 퍼즐 바깥의 테
두리
                g.setColor(255, 255, 0);
                g.fillRect(10, 15, 78, 78);

                puzzle_num = engine.getPuzzleNum();
                timego = true;
                timer(times);
                drawCount();

                for(int i=0; i < sizeMAX; i++){                         // 3X3 모드
                        for(int j=0; j < sizeMAX; j++){
                                drawCell(25 * i +12, 25 * j +17, puzzle_num[i][j]);
                        }
                }
                repaint();
        }
```

```
	// 정해준 위치에 크기에 따라 셀을 그림.
	public void drawCell(int x, int y, int num)
	{
		switch(num)
		{
			case 1:
				g.drawImage(cell2_1, x, y, 0);
			break;
			case 2:
				g.drawImage(cell2_2, x, y, 0);
			break;
			case 3:
				g.drawImage(cell2_3, x, y, 0);
			break;
			case 4:
				g.drawImage(cell2_4, x, y, 0);
			break;
			case 0:
				g.drawImage(cursor, x, y, 0);
			break;
		}
		repaint();
	}

	// 타이머의 표시
	public void timer(int time)
	{
		g.setColor(220, 255, 172);
		g.fillRect(103, 55, 20, 10);
		g.setColor(0, 0, 0);
		g.drawString(""+(time), 124, 53, Graphics.TOP|Graphics.RIGHT);
	}

	// 움직인 이동횟수의 표시
	public void drawCount()
	{
		g.setColor(220, 255, 172);
		g.fillRect(103, 89, 20, 9);
		g.setColor(0, 0, 0);
		g.drawString(""+(moveCount++),		124,		87,
Graphics.TOP|Graphics.RIGHT);
	}

	// 게임중 메뉴 띄우기
	public void runMenu()
	{
		drawBox(20, 25, 88, 94);
		engine.setMode(engine.RUNMENU);
		g.setColor(255, 255, 255);
		g.drawString("Pause", 64, 38, 16|1);
		g.drawString("Resume", 64, 52, 16|1);
		g.drawString("Score", 64, 66, 16|1);
		g.drawString("Back to main", 64, 80, 16|1);
		g.drawString("Exit", 64, 94, 16|1);
		drawRunCursor();
	}

	// 게임중 메뉴의 커서
```

```
	public void drawRunCursor()
	{
		switch(engine.runSwitch)
		{
			case 1:
				g.drawImage(mcur, 37, 43+(14*0), 0);
				g.drawImage(mcur, 84, 43+(14*0), 0);
			break;
			case 2:
				g.drawImage(mcur, 30, 43+(14*1), 0);
				g.drawImage(mcur, 90, 43+(14*1), 0);
			break;
			case 3:
				g.drawImage(mcur, 35, 43+(14*2), 0);
				g.drawImage(mcur, 85, 43+(14*2), 0);
			break;
			case 4:
				g.drawImage(mcur, 25, 43+(14*3), 0);
				g.drawImage(mcur, 96, 43+(14*3), 0);
			break;
			case 5:
				g.drawImage(mcur, 42, 43+(14*4), 0);
				g.drawImage(mcur, 78, 43+(14*4), 0);
			break;
		}
		repaint(20, 25, 88, 94);
	}


public void run() {
    try {
			while(active)
			{
				thread.sleep(200);
				if(timego == true && ((tc++)%5 == 0)){
					timer(times++);
				}
				repaint();
			}
		} catch(InterruptedException e) {
	    }
	}


	public void commandAction(Command c, Displayable d)
{
		String sel = c.getLabel();
		if(sel.equals("SELECT"))
		{
			if(engine.getMode() == engine.TITLE)
			{
				engine.selectMenu();
			}
			else if(engine.getMode() == engine.LEVEL)
			{
				engine.selectLev();
			}
		}
```

```
		else if(sel.equals("START"))
		{
			removeCommand(cmd1);
			addCommand(cmd4);
			addCommand(cmd3);
			engine.keyPress(engine.SOFT_LEFT);
		}
		else if(sel.equals("MAP"))
		{
			timego = false;
			removeCommand(cmd3);
			removeCommand(cmd4);
			addCommand(cmd2);			// GO!
			addCommand(cmd8);			// NULL
			setCommandListener(this);
			drawShowPuzzle(engine.getStageNumber());
			engine.setMode(engine.MISSION);
		}
		else if(sel.equals("MENU"))
		{
			timego = false;
			removeCommand(cmd3);
			removeCommand(cmd4);
			addCommand(cmd5);
			addCommand(cmd6);
			setCommandListener(this);
			runMenu();
		}
		else if(sel.equals("BACK"))
		{
			removeCommand(cmd5);  // BACK
			removeCommand(cmd6);  // SELECT
			addCommand(cmd3);
			addCommand(cmd4);
			engine.setMode(engine.GAME);
			paintCell();
		}
		else if(sel.equals("GO!"))
		{
			timego = true;
			removeCommand(cmd2);
			removeCommand(cmd8);
			addCommand(cmd3);			// GO!
			addCommand(cmd4);			// NULL
			engine.setMode(engine.GAME);
			paintCell();
		}
		else if(sel.equals("OK!"))
		{
			int snum = engine.getStageNumber();
			if(engine.getMode() == engine.SUCCESS)
			{
				engine.setMode(engine.CLEAR);
				drawClearPic(snum);
			}
			else
			{
				tc = 0;
```

```
                                times = 0;
                                moveCount=0;
                                removeCommand(cmd7);
                                if (snum == 3)                  // 3 번 스테이지 끝내면
게임끝냄.
                                {
                                        cube.gameEnd();
                                }
                                engine.setPuzzle(++snum);
                                engine.setStageNumber(snum);
                                engine.setMode(engine.MISSION);
                                drawShowPuzzle(snum);
                        }
                }

        }


        protected void keyPressed(int keyCode) {
                engine.keyPress(keyCode);
        }


        // 각 단계 클리어시 보여주는 그림들
        public void drawClearPic(int drawMode)
        {
                drawBar();
                timer(times);
                drawCount();

                switch (drawMode)
                {
                        case 1:     // clear 1
                                g.drawImage(clear1, 0, 0, 0);
                        break;

                        case 2:     // clear 2
                                g.drawImage(clear2, 0, 0, 0);
                        break;

                        case 3:     // clear 3
                                g.drawImage(clear3, 0, 0, 0);
                        break;
                }
                repaint();
        }
}
```

9 번째는 3X3 퍼즐을 4X4 퍼즐로 만드는 것입니다. ^^;

수정해야할 모듈은

CubeCanvas.java

- **drawCell()**
- **paintCell()**
- **drawClearPic()**
- **drawShowPuzzle()**
- **drawMove()**

CubeEngine.java

- **setPuzzle()**
- **setPuzzleNum()**

CSound.java

```
package Cube;

import com.skt.m.*;
import java.io.*;
import java.util.*;

public class CSound
{
	public final static int SOUND_NUM   = 1;	// 사운드 파일의 개수.
	public final static int STOP		= 10;

	AudioClip clip;
	byte buffer[][] = new byte[SOUND_NUM][];		// 사운드 파일이 많아질 경우를 대비해 배열로 잡음.

	//사운드 로딩 및 생성
	public CSound()
	{
		try {
			clip = AudioSystem.getAudioClip("mmf");
			for (int i=0; i<SOUND_NUM; i++) {
				InputStream is = getClass().getResourceAsStream("/Cube/snd/sound" + i + ".mmf");
				buffer[i] = new byte[is.available()];
				is.read(buffer[i]);
			}
		}catch (Exception ex) {
		}
	}

	// 사운드 플레이
	// playSound(번호) 형식으로 호출하면 해당번호 사운드 재생
	// playSound(STOP) - 사운드 출력정지
	public synchronized void playSound(int num) {
		try {
			if(num != STOP){
				clip.open(buffer[num], 0, buffer[num].length);
				clip.play();
			}else{
				clip.stop();
			}
		}catch (Exception ex) {
		}finally {
			try	{
				clip.close();
			}catch(Exception ex2) {
			}
		}
	}
}
```

CubeCanvas.java

```
package Cube;

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

public class CubeCanvas extends Canvas implements Runnable, CommandListener
{
	// 상수선언
	public final static int SCREEN_WIDTH		= 128;
	public final static int SCREEN_LENGTH		= 143;

	// 이미지 변수선언
	static Image title, mt1, mt2, mt3, mt4, mcur, board, clear1, clear2, clear3, bt;

	static Image cell2_1, cell2_2, cell2_3, cell2_4, cursor, success, clscore, logo;
	static Image cell1,	cell2, cell3, cell4, cell5,	cell6, cell7, cell8, cursor2;


	// 참조클래스 선언
	private CubeCube		cube;
	public	CubeEngine			engine;			// 게임의 로직부분을 담당하는 클래스를 하나더 생성함.

	// 변수선언
	public boolean		active = true;
	public boolean		init = true;
	public boolean		timego;				// 타이머의 진행을 조절하는 On-Off 변수.
	int[][] puzzle_num = new int[4][4];		// 게임중 퍼즐을 저장하는 배열
	int[][] show_puzzle = new int[4][4];		// 미션 퍼즐을 저장하는 배열
	public int			moveCount;			// 커서의 움직인 횟수를 카운드하는 변수.
	public int			sizeMAX;			// 셀의 크기를 결정 (3x3, 4x4)
	public int			times;				// 타이머 누적시간 변수
	private int			tc;
	// 주기적 repaint와 타이머를 함께 쓰기위한 변수.
	private Thread		thread;
	static Graphics		g;


	// Command 버튼선언
	private Command cmd1 = new Command("START", 2, 0);
	private Command cmd2 = new Command("GO!", 4, 0);
	private Command cmd3 = new Command("MAP", 2, 0);
	private Command cmd4 = new Command("MENU", 8, 1);
	private Command cmd5 = new Command("BACK", 2, 0);
	private Command cmd6 = new Command("SELECT", 4, 1);
	private Command cmd7 = new Command("OK!", 4, 1);
	private Command cmd8 = new Command("", 3, 0);		// NULL


	// Constructor
	public CubeCanvas(CubeCube cube)
```

```
	{
		this.cube = cube;
		engine = new CubeEngine(this);
		engine.setCube(cube);
		loadImage();
		thread = new Thread(this);
		thread.start();
	}


	protected void paint(Graphics g)
	{
		this.g = g;
		if(init){
			g.drawImage(logo, 0, 22, 0);
			init = false;
		}
	}

	// 메인메뉴 띄우기
	public void showMainMenu()
	{
		drawBox(20, 25, 88, 94);
		g.setColor(255, 255, 255);
		g.drawString("Start Game", 64, 38, 16|1);
		g.drawString("Instruction", 64, 52, 16|1);
		g.drawString("Credits", 64, 66, 16|1);
		g.drawString("Settings", 64, 80, 16|1);
		g.drawString("Exit", 64, 94, 16|1);
		addCommand(cmd6);
		setCommandListener(this);
		drawCursor();
		repaint();
		serviceRepaints();
	}


	// 메인메뉴 선택커서표시
	public void drawCursor()
	{
		switch(engine.menuSwitch)
		{
			case 1:
				g.drawImage(mcur, 27, 43+(14*0), 0);
				g.drawImage(mcur, 94, 43+(14*0), 0);
			break;
			case 2:
				g.drawImage(mcur, 25, 43+(14*1), 0);
				g.drawImage(mcur, 97, 43+(14*1), 0);
			break;
			case 3:
				g.drawImage(mcur, 35, 43+(14*2), 0);
				g.drawImage(mcur, 86, 43+(14*2), 0);
			break;
			case 4:
				g.drawImage(mcur, 32, 43+(14*3), 0);
				g.drawImage(mcur, 89, 43+(14*3), 0);
			break;
```

```
                        case 5:
                                g.drawImage(mcur, 42, 43+(14*4), 0);
                                g.drawImage(mcur, 78, 43+(14*4), 0);
                        break;
                }
        }


        // 넘겨준 크기값으로 뒷배경사각형을 그리는 부분
        public void drawBox(int x, int y, int w, int h)
        {
                g.setColor(0, 0, 0);
                g.fillRect(x, y, w, h);
                g.setColor(0, 109, 85);
                g.fillRect(x+3, y+3, w-6, h-6);
                g.setColor(255, 219, 0);
                g.drawRect(x+1, y+1, w-3, h-3);
                g.drawImage(mt1, x, y, 0);
                g.drawImage(mt2, x+w-15, y, 0);
                g.drawImage(mt3, x, y+h-16, 0);
                g.drawImage(mt4, x+w-15, y+h-16, 0);
        }

        // 이미지 로딩
        public void loadImage()
        {
                try
               {
                        mt1 = Image.createImage("/Cube/img/m01.png");
                        mt2 = Image.createImage("/Cube/img/m02.png");
                        mt3 = Image.createImage("/Cube/img/m03.png");
                        mt4 = Image.createImage("/Cube/img/m04.png");
                        mcur = Image.createImage("/Cube/img/mcur.png");
                        board    = Image.createImage("/Cube/img/board.png");
                        clear1   = Image.createImage("/Cube/img/clear1.png");
                        clear2   = Image.createImage("/Cube/img/clear2.png");
                        clear3   = Image.createImage("/Cube/img/clear3.png");
                        bt                = Image.createImage("/Cube/img/bt.png");
                        cell2_1  = Image.createImage("/Cube/img/cell2-1.png");
                        cell2_2  = Image.createImage("/Cube/img/cell2-2.png");
                        cell2_3  = Image.createImage("/Cube/img/cell2-3.png");
                        cell2_4  = Image.createImage("/Cube/img/cell2-4.png");
                        cursor   = Image.createImage("/Cube/img/cursor.png");
                        success = Image.createImage("/Cube/img/success.png");
                        clscore = Image.createImage("/Cube/img/cl-score.png");
                        cell1    = Image.createImage("/Cube/img/cell-1.png");
                        cell2    = Image.createImage("/Cube/img/cell-2.png");
                        cell3    = Image.createImage("/Cube/img/cell-3.png");
                        cell4    = Image.createImage("/Cube/img/cell-4.png");
                        cell5    = Image.createImage("/Cube/img/cell-5.png");
                        cell6    = Image.createImage("/Cube/img/cell-6.png");
                        cell7    = Image.createImage("/Cube/img/cell-7.png");
                        cell8    = Image.createImage("/Cube/img/cell-8.png");
                        cursor2  = Image.createImage("/Cube/img/cursor2.png");

                }catch(IOException ex){
                        System.out.println("File not find");
                }
```

```
	}


	//레벨선택 메뉴 그리기
	public void drawLevelMenu()
	{
		drawBox(25, 35, 78, 72);
		g.setColor(255, 255, 255);
		g.drawString("[Level 1]", 65, 48, 16|1);
		g.drawString("[Level 2]", 65, 64, 16|1);
		g.drawString("[Level 3]", 65, 80, 16|1);
		drawCursor2();
		addCommand(cmd8);
		addCommand(cmd6);
		setCommandListener(this);
		repaint();
		serviceRepaints();
	}

	// 레벨메뉴 선택커서
	public void drawCursor2()
	{
		switch(engine.levelSwitch)
		{
			case 1:
				g.drawImage(mcur, 30, 53+(16*0), 0);
				g.drawImage(mcur, 94, 53+(16*0), 0);
			break;
			case 2:
				g.drawImage(mcur, 30, 53+(16*1), 0);
				g.drawImage(mcur, 94, 53+(16*1), 0);
			break;
			case 3:
				g.drawImage(mcur, 30, 53+(16*2), 0);
				g.drawImage(mcur, 94, 53+(16*2), 0);
			break;
		}
	}

	// 화면오른편 게이지보드
	public void drawBar()
	{
		g.setColor(252, 218, 4);
		g.drawLine(97, 0, 97, 108);
		g.drawImage(board, 98, 0, 0);
		g.setColor(220, 255, 172);
		g.fillRect(109, 20, 10, 9);
		g.setColor(0, 0, 0);
		g.drawString(""+(engine.getStageNumber()), 118, 18,
Graphics.TOP|Graphics.RIGHT);
		repaint();
	}


	// 미션제시화면의 표시
	public void drawShowPuzzle(int stageNum)
	{
		drawBar();
```

```
                        // 오른편 점수표 그리기
                drawBackImg();
                // 배경이미지 그리기
                g.setColor(0, 0, 0);
                timer(times);
                drawCount();

                if(engine.getMode() != engine.GAME)
                {
                        removeCommand(cmd6);
                        removeCommand(cmd8);
                        addCommand(cmd1);
                        setCommandListener(this);
                }
                drawBox(18, 8, 93, 115);
                show_puzzle = engine.getShowPuzzle();

                if(stageNum <= 3)
                {
                        sizeMAX = 3;
                        for(int i=0; i < sizeMAX; i++){
                                for(int j=0; j < sizeMAX; j++){
                                        drawCell(25 * i +28 , 25 * j +27,
show_puzzle[j][i]);
                                }
                        }
                }else{
                        sizeMAX = 4;
                        for(int i=0; i < sizeMAX; i++){
                                for(int j=0; j < sizeMAX; j++){
                                        drawCell(19 * i +27 , 19 * j +27,
show_puzzle[j][i]);
                                }
                        }
                }

                g.setColor(255, 255, 255);
                g.drawString("YOUR MISSION", 28, 14, 0);
                g.drawString("STAGE "+stageNum, 43, 103, 0);
                repaint();
        }

        // 뒷배경 이미지 그리기
        public void drawBackImg()
        {
                g.drawImage(bt, 0, 109, 0);
                switch(engine.getStageNumber())
                {
                        case 1:
                                g.drawImage(clear1, 0, 0, 0);
                        break;
                        case 2:
                                g.drawImage(clear2, 0, 0, 0);

                        break;
                        case 3:
                                g.drawImage(clear3, 0, 0, 0);
                        break;
```

```
                        case 4:
                                g.drawImage(clear1, 0, 0, 0);
                        break;
                        case 5:
                                g.drawImage(clear2, 0, 0, 0);

                        break;
                        case 6:
                                g.drawImage(clear3, 0, 0, 0);
                        break;
                }
        }

        // 이동시킨 셀 그리기
        public void drawMove(int cX, int cY, int oX, int oY, int size, int swap)
        {
                if(sizeMAX == 3)
                {
                        g.drawImage(cursor, cX*(size+1)+12, cY*(size+1)+17, 0);
                        drawCell(oX*(size+1)+12, oY*(size+1)+17, swap);
                }else{
                        g.drawImage(cursor2, cX*(size+1)+11, cY*(size+1)+16, 0);
                        drawCell(oX*(size+1)+11, oY*(size+1)+16, swap);
                }
                drawCount();
                if((cX == 2 && cY == 2) || (cX == 3 && cY == 3))                //    특정
위치에 왔을때만 Bingo 검사
                {
                        if (engine.isBingo())
                        {
                                engine.setMode(engine.SUCCESS);
                                drawClear();
                        }
                }
                repaint();
        }


        // SUCCESS 메세지 출력
        public void drawClear()
        {
                removeCommand(cmd3);
                removeCommand(cmd4);
                addCommand(cmd8);                // NULL
                addCommand(cmd7);                // OK!
                setCommandListener(this);

                timego = false;
                g.drawImage(success, 10, 26, 0);

                g.setColor(0, 0, 0);
                g.fillRect(16, 83, 67, 15);
                g.setColor(252, 218, 172);
                g.fillRect(17, 84, 65, 13);

                g.setColor(255, 0, 0);
                g.drawImage(clscore, 20, 86, 0);
                g.drawString(""+engine.calculateScore(),                80,                84,
```

```
Graphics.TOP|Graphics.RIGHT);

                if(engine.getStageNumber() == 6) gameClear();                   // 6 판째
가 마지막. 메세지 보여주기
        }


        // 마지막 스테이지임을 알림
        public void gameClear()
        {
                g.setColor(0, 0, 0);
                g.fillRect(9, 63, 80, 20);
                g.setColor(255, 255, 0);
                g.fillRect(10, 64, 78, 18);
                g.setColor(0, 0, 0);
                g.drawString("Game Cleared", 16, 66, 20);
        }

        // 본게임 화면의 표시
        public void paintCell()
        {
                drawBar();
                        // 왼쪽 점수표 그리기
                drawBackImg();
                // 배경이미지 그리기
                g.setColor(0, 0, 0);
                g.drawRect(9, 14, 79, 79);                                      //  퍼즐
바깥의 테두리
                g.setColor(255, 255, 0);
                g.fillRect(10, 15, 78, 78);

                puzzle_num = engine.getPuzzleNum();
                timego = true;
                timer(times);
                drawCount();

                if(engine.getStageNumber() <= 3){                       // 3X3 모드
                        for(int i=0; i < sizeMAX; i++){
                                for(int j=0; j < sizeMAX; j++){
                                        drawCell(25  *  i  +12,  25  *  j  +17,
puzzle_num[i][j]);
                                }
                        }
                }else{
                        // 4X4 모드
                        for(int i=0; i < sizeMAX; i++){
                                for(int j=0; j < sizeMAX; j++){
                                        drawCell(19  *  i  +11,  19  *  j  +16,
puzzle_num[i][j]);
                                }
                        }
                }
                repaint();
        }


        // 정해준 위치에 크기에 따라 셀을 그림.
        public void drawCell(int x, int y, int num)
```

```
	{
		switch(num)
		{
			case 1:
				if(sizeMAX == 3){
					g.drawImage(cell2_1, x, y, 0);
				}else{
					g.drawImage(cell1, x, y, 0);
				}
			break;
			case 2:
				if(sizeMAX == 3){
					g.drawImage(cell2_2, x, y, 0);
				}else{
					g.drawImage(cell2, x, y, 0);
				}
			break;
			case 3:
				if(sizeMAX == 3){
					g.drawImage(cell2_3, x, y, 0);
				}else{
					g.drawImage(cell3, x, y, 0);
				}
			break;
			case 4:
				if(sizeMAX == 3){
					g.drawImage(cell2_4, x, y, 0);
				}else{
					g.drawImage(cell4, x, y, 0);
				}
			break;
			case 5:
					g.drawImage(cell5, x, y, 0);
			break;
			case 6:
					g.drawImage(cell6, x, y, 0);
			break;
			case 7:
					g.drawImage(cell7, x, y, 0);
			break;
			case 8:
					g.drawImage(cell8, x, y, 0);
			break;
			case 0:
				if(sizeMAX == 3){
					g.drawImage(cursor, x, y, 0);
				}else{
					g.drawImage(cursor2, x, y, 0);
				}
			break;
		}
		repaint();
	}


	// 타이머의 표시
	public void timer(int time)
	{
```

```
                g.setColor(220, 255, 172);
                g.fillRect(103, 55, 20, 10);
                g.setColor(0, 0, 0);
                g.drawString(""+(time), 124, 53, Graphics.TOP|Graphics.RIGHT);
        }


        // 움직인 이동횟수의 표시
        public void drawCount()
        {
                g.setColor(220, 255, 172);
                g.fillRect(103, 89, 20, 9);
                g.setColor(0, 0, 0);
                g.drawString(""+(moveCount++),                  124,                  87,
Graphics.TOP|Graphics.RIGHT);
        }

        // 게임중 메뉴 띄우기
        public void runMenu()
        {
                drawBox(20, 25, 88, 94);
                engine.setMode(engine.RUNMENU);
                g.setColor(255, 255, 255);
                g.drawString("Pause", 64, 38, 16|1);
                g.drawString("Resume", 64, 52, 16|1);
                g.drawString("Score", 64, 66, 16|1);
                g.drawString("Back to main", 64, 80, 16|1);
                g.drawString("Exit", 64, 94, 16|1);
                drawRunCursor();
        }

        // 게임중 메뉴의 커서
        public void drawRunCursor()
        {
                switch(engine.runSwitch)
                {
                        case 1:
                                g.drawImage(mcur, 37, 43+(14*0), 0);
                                g.drawImage(mcur, 84, 43+(14*0), 0);
                        break;
                        case 2:
                                g.drawImage(mcur, 30, 43+(14*1), 0);
                                g.drawImage(mcur, 90, 43+(14*1), 0);
                        break;
                        case 3:
                                g.drawImage(mcur, 35, 43+(14*2), 0);
                                g.drawImage(mcur, 85, 43+(14*2), 0);
                        break;
                        case 4:
                                g.drawImage(mcur, 25, 43+(14*3), 0);
                                g.drawImage(mcur, 96, 43+(14*3), 0);
                        break;
                        case 5:
                                g.drawImage(mcur, 42, 43+(14*4), 0);
                                g.drawImage(mcur, 78, 43+(14*4), 0);
                        break;
                }
                repaint(20, 25, 88, 94);
```

```
        }


    public void run() {
        try {
                        while(active)
                        {
                                thread.sleep(200);
                                if(timego == true && ((tc++)%5 == 0)){
                                        timer(times++);
                                }
                                repaint();
                        }
                } catch(InterruptedException e) {
            }
        }


        public void commandAction(Command c, Displayable d)
    {
                String sel = c.getLabel();
                if(sel.equals("SELECT"))
                {
                        if(engine.getMode() == engine.TITLE)
                        {
                                engine.selectMenu();
                        }
                        else if(engine.getMode() == engine.LEVEL)
                        {
                                engine.selectLev();
                        }
                }
                else if(sel.equals("START"))
                {
                        removeCommand(cmd1);
                        addCommand(cmd4);
                        addCommand(cmd3);
                        engine.keyPress(engine.SOFT_LEFT);
                }
                else if(sel.equals("MAP"))
                {
                        timego = false;
                        removeCommand(cmd3);
                        removeCommand(cmd4);
                        addCommand(cmd2);               // GO!
                        addCommand(cmd8);               // NULL
                        setCommandListener(this);
                        drawShowPuzzle(engine.getStageNumber());
                        engine.setMode(engine.MISSION);
                }
                else if(sel.equals("MENU"))
                {
                        timego = false;
                        removeCommand(cmd3);
                        removeCommand(cmd4);
                        addCommand(cmd5);
                        addCommand(cmd6);
                        setCommandListener(this);
```

```
                        runMenu();
                }
                else if(sel.equals("BACK"))
                {
                        removeCommand(cmd5);  // BACK
                        removeCommand(cmd6);  // SELECT
                        addCommand(cmd3);
                        addCommand(cmd4);
                        engine.setMode(engine.GAME);
                        paintCell();
                }
                else if(sel.equals("GO!"))
                {
                        timego = true;
                        removeCommand(cmd2);
                        removeCommand(cmd8);
                        addCommand(cmd3);               // GO!
                        addCommand(cmd4);               // NULL
                        engine.setMode(engine.GAME);
                        paintCell();
                }
                else if(sel.equals("OK!"))
                {
                        int snum = engine.getStageNumber();
                        if(engine.getMode() == engine.SUCCESS)
                        {
                                engine.setMode(engine.CLEAR);
                                drawClearPic(snum);
                        }
                        else
                        {
                                tc = 0;
                                times = 0;
                                moveCount=0;
                                removeCommand(cmd7);
                                if (snum == 6)                  // 6 번 스테이지 끝내면
게임끝냄..
                                {
                                        cube.gameEnd();
                                }
                                engine.setPuzzle(++snum);
                                engine.setStageNumber(snum);
                                engine.setMode(engine.MISSION);
                                drawShowPuzzle(snum);
                        }
                }

        }



        protected void keyPressed(int keyCode) {
                engine.keyPress(keyCode);
        }


        // 각 단계 클리어시 보여주는 그림들
        public void drawClearPic(int drawMode)
```

```
        {
                drawBar();
                timer(times);
                drawCount();

                switch (drawMode)
                {
                        case 1:                         // clear 1
                                g.drawImage(clear1, 0, 0, 0);
                        break;

                        case 2:                         // clear 2
                                g.drawImage(clear2, 0, 0, 0);
                        break;

                        case 3:                         // clear 3
                                g.drawImage(clear3, 0, 0, 0);
                        break;

                        case 4:                         // clear 4
                                g.drawImage(clear1, 0, 0, 0);
                        break;

                        case 5:                         // clear 5
                                g.drawImage(clear2, 0, 0, 0);
                        break;

                        case 6:                         // clear 6
                                g.drawImage(clear3, 0, 0, 0);
                        break;
                }
                repaint();
        }
}
```

CubeCube.java

```
package Cube;

import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import java.io.*;
import com.skt.m.*;

public class CubeCube extends MIDlet
{
        public CubeCanvas  canvas;
        static Graphics g;
        Display display;

        public CubeCube()
    {
                canvas = new CubeCanvas(this);
                Device.setBacklightEnabled(true);          // 키가 눌려질때 마다 백
라이트가 들어오는 기능설정 (기본적으로 true 로 설정)
                Device.enableRestoreLCD(true);             // 통화후 이전화면 복구설
정. SUSPEND 모드에서 돌아올때 화면복구를 위해
                loadLogo();
```

```
        }

        protected void startApp()
    {
                display = Display.getDisplay(this);
          canvas.active = true;
                display.setCurrent(canvas);
                BackLight.on(0);                    // 백라이트 On 으로 설정. 파라미터
0 이면 무한대 켜짐.
    }

    protected void pauseApp()
    {
         canvas.active = false;
    }

    public void resumeApp()
    {
    }

        public void destroyApp(boolean uc)
    {
                display.setCurrent(null);
                notifyDestroyed();
    }

        public void loadLogo()
        {
                try{
                        canvas.logo    =    Image.createImage("/Cube/img/logo.png");

                }catch(IOException e){
                        e.printStackTrace();
                }
        }

        public void gameEnd()
        {
                destroyApp(false);
        }
}
```

CubeEngine.java

```
package Cube;

import javax.microedition.lcdui.*;
import java.util.*;
import java.io.*;
import com.skt.m.*;


public class CubeEngine
{
        // mode 부분 상수들.
        public final static int LOGO                    = 0;
        public final static int TITLE                   = 1;
        public final static int LEVEL                   = 2;
```

```
	public final static int MISSION 			= 3;
	public final static int GAME 		= 4;
	public final static int RUNMENU 			= 5;
	public final static int SUCCESS 			= 6;
	public final static int CLEAR 			= 7;

	// 키입력부분 상수들.
	public final static int KEY_UP 			= 141;
	public final static int KEY_LEFT 		= 142;
	public final static int KEY_RIGHT 		= 145;
	public final static int KEY_DOWN 		= 146;
	public final static int SOFT_LEFT 		= 129;
	public final static int SOFT_RIGHT 		= 131;

	// 멤버 변수
	public int 			cellsize;
	public int 			xySize;
	public int 			cx, cy;
	// 커서의 x, y 값
	public int 			menuSwitch = 1;			// 메뉴
선택을 위해서 쓰이는 모드변수
	public int 			levelSwitch = 1;
	public int 			runSwitch = 1;
	private int 			stageNumber;			// 현재
스테이지의 넘버
	private int 			mode;
	// 화면에 따른 키이벤트를 구분하기 위해서 쓰이는 모드변수.
	private int 			score;
	static int 		swap;					// cell 의
이동시 값 치환을 위한 임시변수
	static int[][] puzzle_num = new int[4][4];  // 퍼즐의 각 픽셀(사각형)에 들어갈
숫자배열
	static int[][] show_puzzle = new int[4][4]; // 미션제시 퍼즐 숫자배열
	String[] stage = new String[6];				//모든 Stage 정
보를 담는 배열


	// 참조클래스 선언
	private CubeCube cube;
	private CubeCanvas	canvas;				//
Canvas
	private CSound	sound;

	// Constructor
	public CubeEngine(CubeCanvas canvas)
	{
		this.canvas = canvas;
		sound = new CSound();
/*
		stage[0] = new String("111112110");
	// 테스트용
		stage[1] = new String("111112110");
		stage[2] = new String("111112110");
		stage[3] = new String("1111111111131110");
		stage[4] = new String("1111111111131110");
		stage[5] = new String("1111111111131110");
*/
		stage[0] = new String("111324110");
```

```
	// 실제포팅용
		stage[1] = new String("131424130");
		stage[2] = new String("424131430");
		stage[3] = new String("1122334411223340");
		stage[4] = new String("5612127863028724");
		stage[5] = new String("3824714678042356");
	}

	// GET & SET method
	public int getMode()
	{
		return mode;
	}

	public void setMode(int mode)
	{
		this.mode = mode;
	}

	public int getStageNumber()
	{
		return stageNumber;
	}

	public void setStageNumber(int num)
	{
		stageNumber = num;
	}

	public int[][] getShowPuzzle()
	{
		return show_puzzle;
	}

	public int[][] getPuzzleNum()
	{
		return puzzle_num;
	}

	public void setCube(CubeCube cube)
	{
		this.cube = cube;
	}



	public void setPuzzle(int stageNum)
	{
		if(stageNum <= 3)
		{
			xySize = 3;
			cellsize=24;
		}else{
			xySize = 4;
			cellsize=18;
		}

		switch (stageNum)
```

```
		{
			case 1:
				setShowNum(3,0);
				setPuzzleNum(3,0);	//  사이즈, 레벨
			break;
			case 2:
				setShowNum(3,1);
				setPuzzleNum(3,1);
			break;
			case 3:
				setShowNum(3,2);
				setPuzzleNum(3,2);
			break;
			case 4:
				setShowNum(4,3);
				setPuzzleNum(4,3);
			break;
			case 5:
				setShowNum(4,4);
				setPuzzleNum(4,4);
			break;
			case 6:
				setShowNum(4,5);
				setPuzzleNum(4,5);
			break;
		}
	}


	public void setShowNum(int size, int level)
	{
		int index=0;
		for(int i=0; i<size; i++){
			for(int j=0; j<size; j++){
				show_puzzle[i][j]                                    =
Integer.parseInt(stage[level].substring(index++,index));
			}
		}
	}


	public void setPuzzleNum(int size, int level)
	{
		int random = 0;
		if(size == 3){
			random = getRandomInt(8);		// 0-7 랜덤출력
		}else{
			random = getRandomInt(15);			//  0-14  랜덤출
력
		}
		int ran = 1;
		for(int i=0; i<size; i++){
			for(int j=0; j<size; j++){
				if (random > size*size-1) random = 0;
				ran                                                  =
Integer.parseInt(stage[level].substring(random++,random));
				if (ran==0) {cx=i; cy=j;}                    //  커서
위치 초기화
```

```
                                puzzle_num[i][j] = ran;
                        }
                }
        }

        static int getRandomInt(int limit) {          // 랜덤하게 숫자 뽑기. limit 이내.
                Random rnd = new Random();
                int number = rnd.nextInt();
                number = (number >>> 16) & 0xffff;
                number /= (0xffff/ limit);
                return number;
        }

        public void showPuzzle()
        {
                mode = MISSION;
                canvas.drawShowPuzzle(stageNumber);
        }

        public void gameStart()
        {
                System.gc();
                canvas.paintCell();
        }

        // 퍼즐셀의 미션완료 검사 (성공하면 true 리턴)
        public boolean isBingo()
        {
                int nCount = 0;
                for (int j=0; j < canvas.sizeMAX; j++){
                        for (int i=0; i < canvas.sizeMAX; i++){
                                if ( puzzle_num[i][j] != show_puzzle[j][i]){
                                        return false;
                                }
                        }
                }
                return true;
        }

        // 가중치를 부여한 점수계산
        public int calculateScore()
        {
                int valuePoint = 0;
                switch(stageNumber)
                {
                        case 1:
                                valuePoint = 25;
                        break;
                        case 2:
                                valuePoint = 12;
                        break;
                        case 3:
                                valuePoint = 10;
                        break;
                        case 4:
                                valuePoint = 6;
                        break;
                        case 5:
```

```
                                valuePoint = 4;
                        break;
                        case 6:
                                valuePoint = 2;
                        break;
                }
                score = (800 - canvas.times -(canvas.moveCount*valuePoint))*10;
                // 점수계산.
                if(score < 100){score = 100;}                   // 최하 100 점.
                return score;
        }



        public void keyPress(int key)
        {
                switch (mode)
                {
                        case LOGO:                                      // Logo
                        case TITLE:
                                keyPressInTitle(key);
                        break;

                        case LEVEL:                                     // Show Level
                                keyPressInLevel(key);
                        break;

                        case MISSION:                           // ShowPuzzle
                                keyPressInShowPuzzle(key);
                        break;

                        case GAME:                                      // In Game
                                keyPressInGame(key);
                        break;

                        case RUNMENU:                           // Game Menu
                                keyPressInRunMenu(key);
                        break;

                        case SUCCESS:                           // Stage Clear
                                keyPressInClear(key);
                        break;

                        case CLEAR:                                     //  Show  Clear
Picture
                                keyPressInClearPicture(key);
                        break;

                }
        }



        public void keyPressInTitle(int key)
        {
                if(mode == LOGO)
                {
                        canvas.showMainMenu();
```

```
                        mode = TITLE;
                        return;
                }
                if ((menuSwitch != 1) && (key == KEY_UP))    // 위방향
                {
                        menuSwitch--;
                        canvas.showMainMenu();
                        sound.playSound(0);
                // 사운드파일 번호
                }
                else if ((menuSwitch != 5) && (key == KEY_DOWN))  // 아래방향
                {
                        menuSwitch++;
                        canvas.showMainMenu();
                        sound.playSound(0);
                // 사운드파일 번호
                }
        }


        public void selectMenu()
        {
                switch (menuSwitch)                          // 현재의 메뉴커서 위치를
가지고 감.
                {
                        case 1:
                                canvas.drawLevelMenu();          // 레벨메뉴 보이
기.
                                mode = LEVEL;
                        break;

                        case 2:
                                // 게임설명 (Instruction)
                                // 이부분은 레벨메뉴를 띄우는 것과 동일한 방식으로
작성 됨
                        break;

                        case 3:
                                // 저작권 정보 (Credits)
                                // 이부분은 레벨메뉴를 띄우는 것과 동일한 방식으로
작성 됨.
                        break;

                        case 4:
                                // 환경설정 (Settings)
                                // 이부분은 레벨메뉴를 띄우는 것과 동일한 방식으로
작성 됨.
                        break;

                        case 5:
                                cube.gameEnd();          // 게임종료 (Exit)
                        break;
                }
        }



        public void keyPressInLevel(int key)
```

```
	{
		if ((levelSwitch != 1) && (key == KEY_UP))		// 위 방
향
		{
			levelSwitch--;
			canvas.drawLevelMenu();
			sound.playSound(0);
				// 사운드 출력
		}
		else if ((levelSwitch != 3) && (key == KEY_DOWN))	// 아래방향
		{
			levelSwitch++;
			canvas.drawLevelMenu();
			sound.playSound(0);
		}
	}


	public void selectLev()
	{
		canvas.times = 0;
		switch (levelSwitch)		// 현재의 선택레
벨을 가지고 감.
		{
			case 1:
	 // Level 1 선택.
				setStageNumber(1);
				setPuzzle(1);
				showPuzzle();
			break;

			case 2:
				// 추가형식.
			break;

			case 3:
				// 추가형식.
			break;
		}
	}


	public void keyPressInShowPuzzle(int key)
	{
		if (key == SOFT_LEFT)		// 왼쪽소프트
		{
			mode = GAME;
			gameStart();
		}
	}


	public void keyPressInGame(int key)
	{
		if(key == SOFT_LEFT){
			canvas.timego = false;
			showPuzzle();		// 게임
중 미션퍼즐 보기
```

```
                }else{
                        switch (key)
                        {
                                case KEY_UP:                            //  위방
향키
                                        if (cy > 0) {
                                                swap                    =
puzzle_num[cx][cy];
                                                puzzle_num[cx][cy]      =
puzzle_num[cx][cy-1];
                                                cy--;
                                                puzzle_num[cx][cy]      =
swap;
                                                canvas.drawMove(cx,   cy,
cx, cy+1, cellsize, puzzle_num[cx][cy+1]);
                                        }
                                break;

                                case KEY_DOWN:                          // 아래
                                        if (cy < xySize-1) {
                                                swap                    =
puzzle_num[cx][cy];
                                                puzzle_num[cx][cy]      =
puzzle_num[cx][cy+1];
                                                cy++;
                                                puzzle_num[cx][cy]      =
swap;
                                                canvas.drawMove(cx,   cy,
cx, cy-1, cellsize, puzzle_num[cx][cy-1]);
                                        }
                                break;

                                case KEY_LEFT:                          // 왼쪽
                                        if (cx > 0) {
                                                swap                    =
puzzle_num[cx][cy];
                                                puzzle_num[cx][cy]      =
puzzle_num[cx-1][cy];
                                                cx--;
                                                puzzle_num[cx][cy]      =
swap;
                                                canvas.drawMove(cx,   cy,
cx+1, cy, cellsize, puzzle_num[cx+1][cy]);
                                        }
                                break;

                                case KEY_RIGHT:                         //  오른
쪽
                                        if (cx < xySize-1) {
                                                swap                    =
puzzle_num[cx][cy];
                                                puzzle_num[cx][cy]      =
puzzle_num[cx+1][cy];
                                                cx++;
                                                puzzle_num[cx][cy]      =
swap;
                                                canvas.drawMove(cx,   cy,
cx-1, cy, cellsize, puzzle_num[cx-1][cy]);
```

```
                                        }
                                break;
                        }
                }
        }


        public void keyPressInRunMenu(int key)
        {
                if ((runSwitch != 1) && (key == KEY_UP))                // 위방향
                {
                        runSwitch--;
                        canvas.runMenu();
                }
                else if((runSwitch != 5) && (key == KEY_DOWN))   // 아래방향
                {
                        runSwitch++;
                        canvas.runMenu();
                }
        }

        public void keyPressInClear(int key)
        {
                if (key == SOFT_RIGHT)              // 오른쪽 소프트
                {
                        mode = CLEAR;
                        canvas.drawClearPic(stageNumber);
                }
        }

        public void keyPressInClearPicture(int key)
        {
        }
}
```

우와 드디어 끝냈습니다. 진짜 길었습니다. 제가 한일은 거의 없지만..여튼 끝내니깐..좋군요.. 당분간 강좌를 쓸수 있을지 모르겠네요..여튼 새로운 좋은 정보 혹은 기법을 터득하면 함 강좌를 올리도록 하겠습니다. 그때까지 다들 좋은 프로그래머가 되시길..

프로그램을 제 방식으로 손댈 수 있지만 여러분의 학습상 도움이 될 것 같아서 일단은 접었습니다. 사실은 프로그램 고치기 무지 귀찮더군요 ㅡㅡ; 그냥 올립니다. 이 소스는 **Neoco** 님의 저작권이 달린 소스이니 소스를 사용시에는 **Neoco** 님의 허락을 받으셔야 할겁니다.
강좌는 계속 이어집니다~~~~~

그럼 즐거운 하루 되십시오..핑크레드..
참고로 **Neoco** 님 또한 소심한 면이 있습니다. ㅋㅋㅋ 자칭 Early Adopter 라고 하시네요..

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.