iアプリ簡易リファレンス　ver0.1.5.1

１．iアプリJava独自のメソッド

（１）iアプリの命令を使えるようにする

```
import com.nttdocomo.ui.*;
```

（２）乱数を使う

```
import java.util.Random;
…
class クラス名 extends Canvas{
        int △;
        Random ◇=new Random();
        …
        public void paint(Graphics g){
                …
                △=Math.abs(◇.nextInt()%☆);
                …
        }
}
```

☆・・・０～☆まで乱数を発生させます

（３）機種ごとの縦横幅を調べる

```
class クラス名 extends Canvas{
        int △,◇;
        public クラス名(){
                △=getWidth();
                ◇=getHeight();
        }
}
```

（４）ソフトキーに名前をつける

```
        public クラス名(){
                setSoftLabel(Frame.SOFT_KEY_1,"名前１");
                setSoftLabel(Frame.SOFT_KEY_2,"名前２");
        }
```

（５）iアプリを実行する

```
public class パブリッククラス名 extends IApplication{
        public void start(){
                Display.setCurrent(new クラス名());
        }
}
```

※パブリッククラス名は、～.java　というファイル名と同じでなければならない

（６）ソフトキーの反応を調べる

```
        public void processEvent(int t,int p){
                if(t==Display.KEY_PRESSED_EVENT && p==Display.KEY_SOFT2){
                        処理１;}
                if(t==Display.KEY_PRESSED_EVENT && p==Display.KEY_SOFT1){
                        処理２;}
        }
```

（７）iアプリを終了させる

```
IApplication.getCurrentApp().terminate();
```

（８）再描画する

```
                repaint();
```

（９）画面描画する

```
        public void paint(Graphics g){
                g.lock();
                画像インスタンスに色を設定する
                画像インスタンスに描画メソッドで指示する
                g.unlock(true);
        }
```

（１０）文字列を使う

```
        String ◇=new String();
```

（１１）ショートタイマーを使う

```
class クラス名 extends Canvas{
        ShortTimer ◇;
        public クラス名(){
                ◇=ShortTimer.getShortTimer(this,ID番号,ミリ秒,true);
        }
        public void processEvent(int t,int p){
                if(t==Display.TIMER_EXPIRED_EVENT){処理１;}
        }
}
```

※◇start();にてショートタイマーを開始する
※◇.stop();にてショートタイマーを止める

（１２）画像を使う

```
class クラス名 extends Canvas{
        Image ◇;
        public クラス名(){
                MediaImage △=MediaManager.getImage("resource:///ファイル名");
                try{△.use();}catch(Exception e){}
                ◇=△.getImage();
        }
        public void paint(Graphics g){
                g.lock();
                g.drawImage(◇,x座標,y座標);
                g.unlock(true);
        }
```

（１３）画面を塗りつぶす

```
                g.setColor(g.getColorOfRGB(R値,G値,B値));
                g.fillRect(0,0,◇,△);
```

（１４）iメロディを使う

```
class クラス名 extends Canvas{
        MediaSound ◇;
        AudioPresenter △;
        public クラス名(){
                △=AudioPresenter.getAudioPresenter();
                ◇=MediaManager.getSound("resource:///iメロディ.MLD");
                try{◇.use();}catch(Exception e){}
                △.setSound(◇);
                △.play();
        }
}
```

※△.stop(); で再生が止まる

（１５）色を設定する

```
                g.setColor(g.getColorOfRGB(R値,G値,B値));
```

（１６）スレッドを利用する

```
class クラス名 extends Canvas implements Runnable{
        boolean ◇=false;
        Thread △=null;
        public クラス名(){
                if(△==null){△=new Thread(this);}
        }
        public void run(){
                while(◇){
                        処理１;
                        try{◇.sleep(ミリ秒);}catch(Exception e){}
                }
                △=null;
                if(△==null){△=new Thread(this);}
        }
```

※◇=true; でスレッドを再開する
※◇=false; でスレッドを停止する
※スレッドはJava2からはwhileなどのループで実現し、stop()メソッドなどは廃止された

（１７）ダイアログを表示させる

```
class クラス名 extends Canvas{
        Dialog ◇=new Dialog(Dialog.BUTTON_OK,"ダイアログボックス名");
        Dialog △=new Dialog(Dialog.DIALOG_YESNO,"イエスノーボックス名");
        public void paint(Graphics g){
                g.lock();
                ◇.setText("メッセージ");
                ◇.show();
                △.setText("メッセージ");
                △.show();
                g.unlock(true);
        }
}
```

（１８）描画メソッド

※スレッドはＪ

①文字列を表示する　　◇.drawString(″文字列″,x座標,y座標);
②画面を消去する　　◇.clearRect(x座標,y座標,横幅,高さ);
　　※消去後は画面は白
③直線を描画する　　◇.drawLine(x1座標,y1座標,x2座標,y2座標);
④四角形の枠を描画する　◇.drawRect(x1座標,y1座標,x2座標,y2座標);
⑤四角形を塗りつぶす　　◇.fillRect(x座標,y座標,横幅,高さ);
⑥多角形の枠を描画する　◇.drawPolyline(配列に入れたx座標,配列に入れたy座標,点の総数);
⑦多角形を塗りつぶす　　◇.fillPolygon(配列に入れたx座標,配列に入れたy座標,点の総数);

※例：drawPolyline fillPolygonは多角形を描くので、座標値を配列に入れておく必要がある。

```
class クラス名 extends Canvas{
        int[] ◇={x1,x2,x3,xn・・・};
        int[] △={y1,y2,y3,yn,・・・};
        public void paint(Graphics g){
                ・・・
                g.fillPolygon(◇,△,配列数);
                ・・・
        }
}
```

２．iアプリの規格

（１）KEY_PRESSED_EVENT時のパラメータに代入される値

KEY0～KEY9	数字キー
KEY_UP	[↑]キー
KEY_DOWN	[↓]キー
KEY_RIGHT	[→]キー
KEY_LEFT	[←]キー
KEY_SELECT	決定キー
KEY_ASTERISK	＊キー
KEY_POUND	＃キー
KEY_SOFT1	ソフトキー１
KEY_SOFT2	ソフトキー２

（２）機種ごとの解像度

F503i	120x130
P503i	120x130
N503i	120x130
SO503	120x120
D503i	132x126

（３）フォントサイズ

F503i	12x12	12x12	12x12	12x12	X	X
P503i	12x12	10x10	12x12	12x12	X	X
N503i	12x12	12x12	12x12	12x12	X	X
SO503	14x12	12x10	14x12	14x12	X	X
D503i	16x16	12x12	16x16	16x16	X	X

３．Java自体の予約語等

（１）条件分岐を使う

```
switch(変数){
        case 0:
        処理１;
        break;
        case 1:
        処理２;
        break;
        default:
        処理３;
}

if(条件１){処理１;}
else if(処理２){処理２;}
```

（２）数値を文字列に変換する

```
String ◇=new String();
・・・
drawString(◇.valueOf(△),10,10);
```

（３）配列を使う

```
class クラス名 extends Canvas{
        String[] ◇={"大吉","中吉","小吉","凶"};
        public void paint(Graphics g){
                g.drawString(◇[変数]0,12);
        }
}

        int[] △={☆,☆,☆,☆,・・・};
```

（４）比較演算子

<	より小さい
>	より大きい
<=	以下
>=	以上
==	等しい
!=	等しくない

（５）代入演算子

=	代入
+=	加算と代入
-=	減算と代入
*=	乗算と代入
/=	除算と代入
++	増加（インクリメント）
--	減少（デクリメント）

（６）コメント

```
//  行末までコメント
/*  ～  */  複数行にまたがるコメント
```

（７）繰り返し

```
while(条件式){
    処理１;
}

for( 初期値 ; 上限 ; 増加量 ){
    処理１;
}
```

（８）変数

boolean	true または flase
byte	-128 ～ 127
short	-32768　32767
int	-2147483648 ～ 2147483647
long	64bit
float	32bit
double	64bit
char	Unicode(¥u0000''～'¥uffff')
void	型がないことを表す

※１　変数名の先頭文字は、英字か＄記号か、アンダーライン（一連の文字で構成する）
※２　変数名に文字数の制限はない
※３　変数名には漢字・ひらがな・カタカナは使えない
※４　変数名にはすでに命令で使用されている、ifやforなど（予約語）は使えない

４．iアプリソースの基本構造

例）CanvasSample1.java

●プロジェクト（ソース全体）
ファイル名と、public class メインクラス名 extends IApplication とがが等しくなければならない

●インポート部
ソース内で利用している命令（メソッドを理解させるための命令郡）を読み込む部分。

```
import com.nttdocomo.ui.*;
import java.util.Random;
```

●パブリッククラス部
このIApplication以下を一番最初に読み込む必須の部分。利用する画面自体をインスタンス化（利用可能に）する部分。必ず内部に、public void start(){・・・}が必要である。

```
public class CanvasSample1 extends IApplication{
        MainCanvas gc;
        public void start(){
        Display.setCurrent(new MainCanvas(););
        }
}
```

●処理部（サブクラス）
具体的に処理をする命令郡（クラスなどのまとまり）を格納する部分
パブリッククラス部で定義した、パブリッククラス名と等しくなっていなければならない。public void paint(){・・・}、public void processEvent(){・・・}など具体的実行部分が存在する。

```
class MainCanvas extends Canvas{
```

●グローバル変数宣言部
paint(){}やprocessEvent(){}など、どのメソッド内でも利用する変数などはここで宣言しておく。

●コンストラクタ部　newなどで作るインスタンス等、オブジェクトや変数をまとめて初期化する。
public パブリッククラス名(){・・・}となる

```
public MainCanvas(){
        setSoftLabel(FRAME.SOFT_KEY_2,"終了");
        }
```

●実行部　具体的に、イベント感知やpaint()などのメソッドが並ぶ

```
        static Random rnd = new Random();
        int i;
        public void paint(Graphics g){
                g.lock();
                g.setColor(Graphics.getColorOfName(Graphics.BLACK));
                g.fillRect(0,0,getWidth(),getHeight());
                g.setColor(Graphics.getColorOfName(Graphics.WHITE));
                if(rnd.nextInt()%2==0){g.drawString("Hello",30,12);}
                else{g.drawString("World",30,12);}
                g.unlock(true);
        }
}
```