1　目的

NTT DoCoMo の携帯電話で動作する「i アプリ (i-αppli)」という小型の Java プログラムを作り，Java を使用することでパソコン以外の機器などに対してどのようなことができるかを知る．

2　原理

Java にはサーバ環境向け，デスクトップ環境向け，家電製品・組み込み機器向けの用途別に 3 種類の「エディション (Edtion)」がある．

「J2ME(Java2 Platform,Micro Edition,Connected Limmited Device Configuration) CLDC(Connected, Limited Device Configuration)」は組み込み機器向けのエディションの中の，メモリ容量などに制限のある小型ネットワーク機器用の「コンフィグレーション (Configuration)」になっているものである．これは Palm などでも使われている．

この，J2ME CDLC では，仮想マシンも KVM(K Virtual Machine) とよばれる数百 KB 程度の少ないメモリで動作する専用のものを使用する．

i アプリは，上記の J2ME CLDC と，「MIDP(Mobile Information Device Profile)」，i モード携帯電話用の「i アプリ API」等によって構成されている．このような仕様は「プロファイル (Profile)」といい，i アプリのものには「DoJa2.0」や「DoJa2.1」といった名前がついている．

3　実験方法

3.1　環境

今回の実験では DoJa2.0 向けの i アプリ開発ツール「iαppli Development Kit」を使用した．このツールは NTT DoCoMo のウェブサイト「DoCoMo Net(http://www.nttdocomo.co.jp/)」から入手できる．また，エディタは「notepad」を使用したが，同様の機能があれば他の物でも良い．

3.2　プログラミングの手順

1.「iαppliTool」を起動する．

2. 新規のプロジェクトを作成し，プログラム・ソースを入力しウインドウに表示されている「ソースファイルの場所」に従って保存する．
3. ADF 設定 をクリックし，下のように設定する．
   - AppName: `hellowrld`
   - AppClass: `helloworld.Main`
   - LastModified: (当日の日付，`yyyy.mm.dd` の形式)
4. ビルド をクリックし，エラーがなければ 起動 をクリックし携帯電話のエミュレータ上で実行する．
5. 今回は行っていないが，実際に携帯電話で実行するには，リリース用コンパイルや，JAR ファイルの作成，JAM ファイル（リリースするライブラリ全体を管理する）の修正などの作業が必要である．

4 実験結果

4.1 各リストについて

今回は，最初に，文字表示，画像表示などをするプログラムを使い実験したが，ソースは全員共通であるので，これらのプログラムについてはソースと実行結果は添付しない．

4.2 実習課題

配布されたソースを課題に沿って書き換えたものと，その実行結果を添付する．

4.2.1 文字表示について

1 行の表示文字数は，全角で 13 文字と次の文字の端，半角で 26 文字と次の文字の端が表示された．(実行結果参照) ただし，これは，エミュレータの設定が「device1」の場合で，設定によって画面の大きさは異なる．また，実際の携帯電話でも機種によって，大きな差がある．

また，ラベルの初期化の後 (List 2 参照) に入れた下の 2 つの文は色の設定を変えるためのものとわかった．

```
13 lblHello.setForeground(Graphics.getColorOfRGB(0xff,0xff,0x0));
14 lblHello.setBackground(Graphics.getColorOfRGB(0x0,0x0,0x0));
```

それぞれ，上の文が描画色の，下の文が背景色の設定である．この書き方の場合，色の指定方法は RGB 方式となるので，0x00–0xff または，0–255 の範囲で赤・緑・青 (光の 3 原色) の各色の光について明るさで色を指定している．

4.2.2 GIF ファイルの表示について

画像を表示できる部分の大きさは，約 160×178(単位:ピクセル) だった．

5 考察

5.1 各リストについて

基本的に，新しく出てきた文や方法などにのみ説明をつけた．また，各リストの完全なものは別に添付してあるので，解説の必要のないと思われたものはここには載せていない．なお，左側の数字は行数である．

5.1.1 List 1 : Main.java

```
1 package sayHelloWorld;
```

パッケージは，ここではプロジェクトとほぼ同じ意味で，複数のクラスをまとめて扱うための概念である．クラス・ライブラリとも呼ばれる．

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

i アプリ専用 API のなかのユーザ・インターフェース (UI) 用のライブラリを `import` している．

```
6 public class Main extends IApplication{
```

クラス名は「ADF」の設定によるので，`Main` でなくても良い．

```
7 static IApplication app;
8 static MainPanel mPanel;
```

スーパークラスのインスタンスと，同じパッケージ内にある `MainPanel` のインスタンスを宣言．

```
10 public void start(){
```

start() は継承したメソッドで，i アプリが開始されたときに実行される．(ここではオーバーライドしている．)

```
11 app=this;
12 mPanel=new MainPanel();
```

初期化作業．11 行のように，スーパークラス型インスタンスにはそのサブクラス型のインスタンスを代入できる．正確には，app にすでに実体化されている this の実体のアドレスを入れているだけで，メモリ上のインスタンスは 1 つであり，app は this のポインタのような状態になる．

```
13 Display.setCurrent(mPanel);
```

実際の表示をする部分へ mPanel(のアドレスを) を渡しておくことで，あたかも mPanel が表示内容そのものであるかの様に扱うことが出来る．

5.1.2 List 2 : MainPanel.java

```
6 class MainPanel extends Panel{
```

Panel は GUI 用の部品をまとめて扱うためのクラス．

```
7 Label lblKanji;
8 Label lblEisu;
9 Label lblHello;
```

各ラベルの宣言．

```
11 MainPanel(){
```

クラス名と同名のメソッドは，インスタンスの作成時に自動的に呼ばれる．

```
12 lblHello=new Label("Hello, world !!");
15 lblKanji=new Label("壱弐参四五六七八九拾一二三四五六");
18 lblEisu=new Label ("1234567890abcdefghijklmnopq");
```

```
13 lblHello.setForeground(Graphics.getColorOfRGB(0xff,0xff,0x0));
14 lblHello.setBackground(Graphics.getColorOfRGB(0x0,0x0,0x0));
16 lblKanji.setForeground(Graphics.getColorOfRGB(0xaf,0xaf,0xff));
17 lblKanji.setBackground(Graphics.getColorOfRGB(0x0,0x0,0x0));
19 lblEisu.setForeground(Graphics.getColorOfRGB(0xff,0x0,0x0));
20 lblEisu.setBackground(Graphics.getColorOfRGB(0x0,0x0,0xff));
```

背景 (Background) の色と描画用 (Foreground) の色を設定.

```
21 this.add(lblHello);
22 this.add(lblKanji);
23 this.add(lblEisu);
```

ラベルをこのクラス自身に載せる.

```
24 this.setSoftKeyListener(new SoftKeyListenerClass());
```

このクラス自身をイベント・ソースに設定する.

```
25 this.setSoftLabel(Frame.SOFT_KEY_1,"Exit");
```

SOFT_KEY_1 は画面左側のプログラム側で用途を決めることが出来るキー．その，用途などが画面下端に表示できるので，これを「Exit」に設定している.

```
28 class SoftKeyListenerClass implements SoftKeyListener{
```

MainPanel で発生したイベントのイベント・リスナとなるクラス.

```
29 public void softKeyPressed(int index){}
```

必要が無いメソッドもインターフェイスに入っているものは書かなければならない.

```
30 public void softKeyReleased(int index){
31 if(index==Frame.SOFT_KEY_1){ Main.app.terminate();}
```

キーが，(押されたあと) 放されたときのためのメソッド．SOFT_KEY_1 が押されたときのみアップレットを終了する.

5.1.3 List 3 : Main.java

```
6 static MainCanvas MCanvas;
10 MCanvas=new MainCanvas();
11 Display.setCurrent(MCanvas);
```

Canvas は画像を表示するためのクラス．ここでは，上記の Panel と同様の扱い方をしている．

5.1.4 List 4 : MainCanvas.java

```
4 import com.nttdocomo.io.ConnectionException;
```

エラー処理に使う．

```
6 class MainCanvas extends Canvas{
```

画像を表示するので，Canvas を継承．

```
7 Image gifImage;
```

画像 (静止画) を入れておくためのインスタンス．MediaImage クラスの getImage() メソッドが戻す型なので，それを入れておくために必要．

```
10 MediaImage mi=MediaManager.getImage("resource:///hyouji.gif");
```

MediaImage はリソースを画像として扱うためのクラスで，スーパークラスは MediaResorce．MediaManager リソースを読みとって MediaResorce 型インスタンス

```
11 try{
12 mi.use();
13 }catch(ConnectionException ce){
14 }catch(UIException uie){
15 }
```

mi にファイルを実際に読みこむため，エラーに備えた書き方をしている．

```
16 gifImage=mi.getImage();
17 setSoftLabel(Frame.SOFT_KEY_1,"Exit");
```

gifImage に mi の中のにある静止画を代入.

```
20 public void paint(Graphics g){
```

最初と，表示を更新するときに呼ばれる.

```
21 g.setOrigin(0,0);
```

描画座標の原点を設定する (移動させる).

```
23 g.drawImage(gifImage,0,0);
```

画面に画像を表示する．引数のうち後ろの 2 つは座標指定である.

```
22 g.lock();
24 g.unlock(true);
```

この 2 つの文の間で行った描画は，すべてその場では表示されず，「オフ・スクリーン」とよばれる場所に描画・保存され，g.unlock(true) メソッドが呼ばれたときに初めて実際の画面である「オン・スクリーン」に表示される．このような操作を「ダブルバッファリング」といい，アニメーションなど複雑な描画を行うときの「ちらつき」を押さえるために使われる.

```
27 public void processEvent(int type,int param){
28  if(type==Display.KEY_RELEASED_EVENT){
29   if(param==Display.KEY_SOFT1){ Main.app.terminate(); }
30  }
31 }
```

5.2 i アプリの可能性など

現在, i アプリの最大容量は JAR の状態で 30KB であり,「スクラッチ・バッド (Scratch Pad)」という情報を保存しておく場所の容量も数十 KB 程度である．よって，プログラムとしてはあまり大きな物を作るのは不可能である．また，セキュリティー上の理由によりそれ自体がダウンロードされたサーバとしか交信が出来ないなどの大きな制限もある.

もともと，i モードがあれば HTTP 経由でデータベースなどは十分使えるので，i アプリに関してはあまりビジネス向きには使われないのではないかと思う．

i アプリ関係のドキュメントを見ていると，基本的な用途は，今まで i モードではスピードが足りなかったり通信料金がかかりすぎるようなゲームや，双方向的な機能をつけた「待ちうけ」や，ある程度状況に応じた反応をする 3D アニメーションなどが考えられているようだ．

実際，良く使われる分野はアミューズメント系のようで，自分が学生である事もあるが，少なくとも知っている範囲ではそれ以外での応用的な使い方は見られない．

将来，携帯電話が現在とは違ったとらえられ方をされるようになれば，現在のパーム・トップ程度の性能が得られるかもしれない．そうすればビジネスなど「ひまつぶし」ではない実用的なプログラムも開発され販売されるようになるのではないかと思うが，やはり，現在の携帯電話の性能ではどうしても最近のような状態が限界なのだろう．

しかし，現在の状態は，考え方を変えれば，1 人でもある程度製品に近いものが作れたり，古いファミコンのソフトウエアなどの古くなったプログラムをもう一度復活させる流れができたり，作る側になれば，いかに資源を有効に使うかを考えたりるようになったりと，これはこれなりに面白い状態だとも思う．

参考文献

[1]『Java の応用—i アプリ—』(配布プリント),
東京都立工業高等専門学校電子情報工学科, 2003/2/9

[2]『DoCoMo Net』(http://www.nttdocomo.co.jp/index.shtml)

[3]『Unofficial “DoCoMo Profile–2.0” (Standard API)API Reference.』
(http://godwood.allnet.ne.jp/vioret/dojaapi2/), さかきけい