Java アプリによる
J-PHONE のアプリ作成

実験期日：2002 年後期～後期中間テスト前

所要時間：40 時間くらい

提出日　：2002/12/5

4 年情報工学科
No. ３４　　山村拓也

１．目的

java 言語を使って、J-PHONE で動く待ち受けアプリを作成する。

２．概要

今回作成するアプリは待ち受けアプリといって、携帯電話上に常駐しバッテリー残量、電波状態を常に表示し、音声着信、メール着信やアラームなどのイベントに対しキャラクターのアニメーションなどの反応を返すアプリである。

３．作成方法

待ち受けアプリの作成には大きく分けて、下のような５つの工程がある。

① プログラムの作成
② Class ファイルの作成
③ Class ファイルの事前検証
④ Manifest ファイルの作成と jar、jad ファイルの作成
⑤ エミュレータでのテスト

①プログラムの作成

今回作成したプログラムは、j2ME と j-phone のアプリ開発に沿って開発した。
プログラムリストは次のようになる。

```
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import com.jblend.media.jpeg.*;
import com.j_phone.midlet.*;
import com.j_phone.system.DeviceControl;
import com.jblend.graphics.sprite.*;
```

```
/**************************************************************************
メインクラス
**************************************************************************/
public class waitAppli extends ResidentMIDlet
{
waitAppliCanvas canvas;

/*========== アプリの開始 ==========*/
public void startApp()
{
//キャンバス
canvas = new waitAppliCanvas();
Display.getDisplay(this).setCurrent(canvas);

ringStopped(); // 初期モード設定
}

/*========== アプリの一時停止 ==========*/
public void pauseApp() {
}

/*========== アプリの終了 ==========*/
public void destroyApp(boolean unconditional) {
}

/*========== 電話がかかってきた時 ==========*/
public void ring(String name, String number)
{
canvas.mode = canvas.MD_TEL;
if ( name != null ) {
canvas.dspMsg1 = name;
} else {
canvas.dspMsg1 = number;
}
canvas.dspMsg2 = "から電話だよ。";
canvas.repaint();
```

```
}

/*========== 着信通知を開始した時 ==========*/
public void ringStarted()
{
}

/*========== 着信通知を終了した時 ==========*/
public void ringStopped()
{
canvas.mode = canvas.MD_WAIT;
canvas.dspMsg1 = "";
canvas.dspMsg2 = "ZZZ…";
canvas.repaint();
}

/*========== アラーム時刻になった時 ==========*/
public void notice(String comment)
{
canvas.mode = canvas.MD_ALARM;
canvas.dspMsg1 = comment;
canvas.dspMsg2 = "だよー！！";
canvas.repaint();
}

/*========== 電話が切れた時 ==========*/
public void ignored()
{
}

/*========== SMS を着信した時 ==========*/
public void received(String name, String address, int detail)
{
String detailName[] = { "ｽｶｲﾒｰﾙ","ﾘﾚｰﾒｰﾙ","ｸﾞﾘｰﾃｨﾝｸﾞ","ｽｰﾊﾟｰﾒｰﾙ","自動配信ﾒｯｾｰｼﾞ","ｽﾃｰｼｮﾝﾒｯｾｰｼﾞ","緊急ｽﾃｰｼｮﾝﾒｯｾｰｼﾞ" };
```

```
canvas.mode = canvas.MD_MAIL;
if ( name != null ) {
canvas.dspMsg1 = name;
} else {
canvas.dspMsg1 = address;
}
canvas.dspMsg2 = "の"+detailName[detail-1]+"だよ。";
canvas.repaint();
}

}

/**************************************************************************
キャンバスクラス
**************************************************************************/
class waitAppliCanvas extends Canvas implements Runnable
{
// モード
final int MD_WAIT = 0, // 待ち受けモード
MD_TEL = 1, // 通話モード
MD_MAIL = 2, // メールモード
MD_ALARM = 3; // アラームモード
Image im[] = new Image[4];
int mode;

int battery; //バッテリー残量
int intensity; //電界強度

boolean keyOnFlag = true; //キー入力許可フラグ

DeviceControl devCtl = DeviceControl.getDefaultDeviceControl(); // デバイスコントロールオブジェクト

String dspMsg1, dspMsg2;
```

```
/*========== コンストラクタ ==========*/
public waitAppliCanvas()
{
mode = MD_WAIT; // 初期モード設定
deviceInfo(); //初期端末状態取得

// グラフィック読み込み
for ( int ix=0 ; ix<4 ; ix++ ) {
try {
im[ix] = Image.createImage("/pic0"+ix+".JPG");
} catch ( Exception e ) {}
}

}

/*========== 実行 ==========*/
public void run()
{

while(true ) {

deviceInfo(); // デバイス状態取得
repaint(); // リペイント

//スリープ
try {
Thread.sleep(100);
}catch (Exception e) {
}

}
}
```

```
/*========== paint メソッド ==========*/
public void paint(Graphics g)
{
int pictNo[] = { 0,1,2,3 };

g.setColor((255<<16)+(255<<8)+255 );
g.fillRect( 0,0, getWidth(), getHeight() );

if ( im[pictNo[mode]] != null ) {
g.drawImage( im[pictNo[mode]], getWidth()/2-48, getHeight()-90, g.TOP|g.LEFT);

deviceInfo(); // デバイス状態取得
repaint(); // リペイント

g.setColor(255,0,0);
g.fillRect( 0,0, +battery, 12 );
g.setColor(0,255,0);
g.fillRect( 0,12, +intensity , 12 );
g.setColor(0,0,0);
g.drawString("バッテリー："+battery,0,0,g.LEFT|g.TOP);
g.drawString("%",100,0,g.LEFT|g.TOP);
g.drawString("電界強度　："+intensity,0,12,g.LEFT|g.TOP);
g.drawString("%",100,12,g.LEFT|g.TOP);
}

g.setColor(0,0,0);
if ( mode != MD_WAIT ) {
g.drawString( dspMsg1, 60, getHeight()-12,g.HCENTER|g.BOTTOM);
}
g.drawString( dspMsg2, 70, getHeight(),g.HCENTER|g.BOTTOM);
}

/*========== デバイス状態取得 ==========*/
public void deviceInfo()
{
battery = devCtl.getDeviceState(devCtl.BATTERY); // バッテリー残量
```

```
intensity = devCtl.getDeviceState(devCtl.FIELD_INTENSITY ); // 電界強度

}

}
```

※ SMS とは音声着信以外のさまざまな情報受信のことである。

②class ファイルの作成

class ファイルというのは①のプログラムを class ごとに圧縮しておくものである。
今回の場合はメインクラス、キャンバスクラスに分けられるので 2 つの class ファイルができる。
その変換方法は下のようになる。（コンパイルは java2 SDK の java コンパイラ（javac）使用する。）

１．javac が入っているフォルダをカレントドライブにする。

2. 次にコンパイルしたい java ファイルを下の {} の中にいれてコマンドを入力します。

```
Javac-bootclasspath c:¥J-PHONE-SDK¥stubclasses.zip {}
```

この式の意味は C ドライブにインストールした J-PHONE エミュレータの「stubclasses」を参照して、その class ファイルを {} に作るという意味になる。
これで class ファイルが作成される。

③class ファイルの事前検証

class ファイルは実はそのままでは使えないので、事前検証というのを使って検証済みの class ファイルにする。

1. カレントドライブを class ファイルが入ってるフォルダにする。（全てのクラスで行う。）
2. 次に java2 SDK の preverify を使って class を検証する。

```
preverify -classpath c:¥J-PHONE-SDK¥stubclasses.zip {}
```

この式の意味は、stubclasses を参照して {} にある class ファイルを検証するということになる。
これで検証が完了して、class ファイルがある場所に ouput フォルダができてその中に検証済みの class が入る。

④Manifest ファイルの作成と jar、jad ファイルの作成

マニフィストファイルとは、アプリケーションの性質を示したファイルでこれと先ほど作った class ファイル
を統合して jar ファイルをつくります。
１．マニフィストファイルの作成

マニフィストファイルはメモ帳を使って作成します。

```
MIDlet-1: waitAppli,,waitAppli              //（アプリの構成説明　名前,アイコン,クラス）
MIDlet-Name: waitAppli                        //（アプリの名称）
MIDlet-Vendor: takuya                         //（ベンダー名）
MIDlet-Version: 1.0                           //（バージョン番号）
MicroEdition-Configuration: CLDC-1.0   //（Configuration 名）
MicroEdition-Profile: MIDP-1.0             //（Profile 名）
```

以下の書式で作成し、最後にファイルの拡張子を.MF にすればマニフィストファイルができる。

2. jar ファイルの作成

jar ファイルは下の書式で作成する。

E:¥jbuilder5¥jdk1.3¥bin¥jar cfm *.jar MANIFEST.MF -C output .**

この***に jar ファイル名を入力すると、このフォルダの output のフォルダの中身と
manifest ファイルを圧縮して jar ファイルを作成する。

3.jad ファイルの作成
jad ファイルもマニフィストファイルと同様に、メモ帳を使って作成する。

```
MIDlet-Name: waitAppli                  //（アプリの名称）
MIDlet-Vendor: takuya                   //（ベンダー名）
MIDlet-Version: 1.0                     //（バージョン番号）
MIDlet-Jar-Size: 1315                   //（Jar ファイルのサイズ）
MIDlet-Jar-URL: waitAppli.jar           //（Jar ファイルの URL）
```

以下の書式で作成し、最後にファイルの拡張子を.jar にすれば jar ファイルができます。

⑤エミュレータでのテスト

１．あらかじめダウンロードした J-SKY Application Emulator を起動する。
２．作成した jad ファイルを開く。
この時点でエラーが出たらまた①に戻って、今までの動作を繰り返していく。

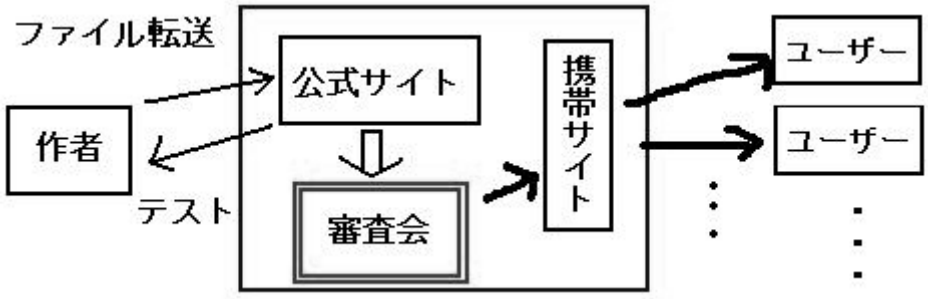
そして出来上がったプログラムがちゃんと動くかどうかを確かめる。

４．配信

配信の手順は下のようになる。
なお配信するためには、フリーではない自分のアドレスが必要となる。

１．公式サイトに登録する。
２．作成した jad、jar ファイルを転送する。
転送することによって、自分だけテストダウンロードができるので実機テストを行う。
これによってデバックができたら、正式にアプリを登録する。
３．審査会を通して配布 。

配信の流れは下の図のようになる。



５．考察

今回の実験で質問が出たのでそれについて考察する。

１．なぜ jad ファイルが必要なのか。

作者が求めているプログラムかどうかを最終的に確かめるために、jad ファイルと jar を比べて確認を取るためである。

２．なぜアプリは自由に配布できないのか。

J-PHONE のアプリは通信機能や携帯内の着信履歴などを情報として取得することが可能なので、悪用される可能性があるため、公式サイトの審査会を通して配布することで安全性を高めているからである。

３．なぜ java 言語なのか。

java 言語はそもそも C 言語などとは違い、ハードに依存しない言語なので携帯などに応用することが可能なためである。

４．大手 3 社のアプリ開発の違いを説明せよ。

まずＤ社の場合は、アプリの開発については 1 番進んでおり、アプリを作ることも簡単になってきている。
しかし個人で勝手に配布できてしまうため、著作権やウィルスに関してはまだまだあまいところがある。

次にＡ社の場合は、1 番遅くアプリに参入したので、あまり資料がなくアプリを作ることも難しい。
しかしＪ社と同じく J2me の規格なので、Ｊ社と同じプログラムを作成することも容易である。
ダウンロードに関しては、特殊な形態をとっており機能制限付なら個人でも配布できる。
しかし、全ての機能を使うなら、サイトを通しての配布となる。

最後に今回開発したＪ社は、開発環境は今回示したとおりである。
ダウンロードに関しては、完全なサイトを通しての配布で個人での配布はできない。
しかし、その分セキュリティ面では強化されており審査会を通らないと正式な配布とならないので、セキュリティは 3 社の中では 1 番高いと思われる。

６．感想

今回は初めての課題実験で、しかも自由課題と言うことなので自分のやりたいことをやってみた。

しかし、アプリ開発は全く未知の領域で始めは何をしていいか、わからなかった。
何とか形にはなったが、最初の思惑とは違いとんだお粗末なアプリになってしまった。
もう少し時間があったらこの他にもいろいろな機能をつけて、完璧な状態にし配布までしたかったが、こんなアプリだったら配布するのが恥ずかしいので今回はやめてしまった。
もし機会があったら続きも作っていきたい。