Core Course-XVIII - 17PCSP06 - LAB - VI – MOBILE AP PLICATION DEVELOPMENT LAB

Credits: 2

Course Objective:

- To understand the concepts and develop the programming skills in J2ME

1. Study of WML and J2ME simulators
2. Design of simple Calculator having +,-,* and / using WML/J2ME
3. Design of Calendar for any given month and year using WML/J2ME
4. Design a Timer to System Time using WML/J2ME
5. Design of simple game using WML/J2ME
6. Animate an image using WML/J2ME
7. Design a personal phone book containing the name, phone no., address, e-mail, etc.
8. Simulation of Authentication and encryption technique used in GSM
9. Browsing the Internet using Mobile phone simulator
10. Study of GlomoSim Simulator

SOFTWARE REQUIREMENTS FOR J2ME PROGRAM

- Nebeans7.0 ml Windows
- Java setup 6.0
- Jdk 6- nb7.0

1.Study of WML and J2ME simulators

Aim:

To study about WML and J2ME simulator.

Java 2 Platform, Micro Edition (J2ME)

Introduction:

Traditional computing devices use fairly standard hardware configurations such as display, keyboard,large amount of memory and permanent storage. However new breed of computing devices lacs hardware configuration. J2ME is specially designed for developing applications for small computing devices such as cell phones, PDA etc.

J2ME Configurations:

Configuration defines the JVM for a particular small computing device.

Types:

Two types of configuration haves been defined.

1) CLDC (Connected limited Device configuration)

CLDC is used for the devices with the limited resources. CLDC devices use stripped version of JVM called KVM. CLDC devices are mobile phones,PDA etc.

2) CDC (Connected device configuration)

CDC devices use complete JVM. CDC devices are set-top box, Home appliances such as Air conditioner etc..

J2ME Profiles:

Profile consists of classes that enable developers to implements features found n a related group of small computing devices.

Many profile are available . Here we use MIDP (Mobile Information Device Profile) MIDP is used with CLDC configuration that provides classes for local storage, a user interface and networking capabilities. Other profiles are Game profile, Foundation profile, RMI profile and many more..

Java Virtual Machine layer:

This layer is an implementation of a Java Virtual Machine that is customized for a particular device's host operating system and supports a particular J2ME configuration.

Configuration layer:

The configuration layer defines the minimum set of Java Virtual Machine features and Java class libraries available on a particular category of devices. In a way, a configuration defines the commonality of the Java platform features and libraries that developers can assume to be available on all devices belonging to a particular category. This layer is less visible to users, but is very important to profile implementers.

Profile layer:

The profile layer defines the minimum set of application programming interfaces (APIs) available on a particular family of devices.

Profiles are implemented upon a particular configuration. Applications are written for a particular profile and are thus portable to any device that supports that profile. A device can support multiple profiles. This is the layer that is most visible to users and application providers.

MIDP layer:

The Mobile Information Device Profile (MIDP) is a set of Java APIs that addresses issues such as user interface, persistence storage, and networking.

MIDlet Programming:

A MIDlet is class , which is controlled by the application manager. A MIDlet class must contain three abstract methods that are called by application manager.

```
public class class-name extends MIDlet
{
public        void
startApp(){      }
public        void
pauseApp() { }
public void destroyApp( unconditional boolean) { }
}
```

Steps for creating a MIDlet suite:

1) Select a file system where you want to create a MIDlet suite. Right click on it, a popup menu appears. From the pop up menu, select New -> MIDlet Suite.
2) Enter the Name of MIDlet suite. Click on Next.
3) It gives you two options.

Allows you to create a new Midlet :

Enter the package and class name . This is the name of a MIDlet that is to be created.

Allows you to add existing MIDlet :

To add existing MIDlet ,click on brows , select the MIDlet to be added. J2ME It also allows you to select icon for the given MIDlet. 4) Click Finish.

ADDING MIDlet to a MIDlet suite:

1) Select a MIDlet suite , to which you want to add Midlet. Right click on it, a pop up menu appears. Select Edit suite from the popup menu.
2) Click on ADD button to add a new MIDlet.
3) Enter name of package and class name.4) Click on OK.

New MIDlet is added to your MIDlet suite.

An example J2ME application using Radio buttons :

Description:

Item class is a base class for a number of derived classes that can be contained within a form class. These derived classes are ChoiceGroup , DateField, ImageItem, StringItem and TextField.

The ChoiceGroup class is used to create check boxes or radio buttons on a form. The state of an instance of a class derived from the Item class changes whenever user enters data into the instance ,such as radio button is selected. An ItemListener monitors the events during a life of the MIDlet and traps events that changes in the state of any Item class contained on a form.

An instance of a ChoiceGrioup class can be of two types: Exclusive or multiple. An exclusive instance appears as a set of radio buttons and a multiple instance contains set of check boxes.

Classes used:

Form Class: It is a container for the other displayable objects that appear on the screen. Any derived class instance of the Item class can be placed on the instance of Form class.

ChoiceGroup class :

J2ME classifies check boxes and radio buttons as the instance of ChoiceGroup class. We set the type of choiceGroup instance to EXCLUSIVE to create radio buttons.

ItemStateListener:

Any MIDlet that utilizes instances of the Item class withon a form must implement ItemStateListener and must have an itemStateChanged() method.

Creating a Radio buttons :

private ChoiceGroup gender;

```
gender = new ChioceGroup (“Enter gender”, Choice.EXCLUSIVE) ;
gender.append(“Female”,null); gender.append(“Male”,null);
```

J2ME program to insert an image in a Canvas.

Description:

There are two types of images that can be displayed.

1) Immutable
2) Mutable

An immutable images are loaded from a file or other resources and cannot be modified once the image is displayed. Examples. Icon s associated with MIDlets

are immutable.

A mutable image is drawn on the Canvas using methods available in Graphics class.

An immutable image can be drawn on the screen as well as on Canvas. where as mutable image is drawn only on the canvas.

The Canvas Class:

Each MIDlet has a one instance of the Display class, and the Display class has a one derived class called 'displayable'. Everything a MIDlet displays on the Screen is created by an instance of a Displayable class.

The Display class hierarchy is shown below:

```
public class Displayable public abstract
class Displayable public abstract class
Screen extends Displayable public abstract
class Canvas extends Displayable
public class Graphics
```

The screen class is used to create high-level components. the Canvas class is used to gain lowlevel access to display.

Creating an Image using Image class:

Create an instance of an Image class by calling the createImage() method of the Image class.

```
private Image img;
img =
Image.createImage
(path);
```

Displaying an Image on a Canvas

Once the image is created , it can be drawn on the Canvvs using drawImage() method of the Graphics class. Graphics.drawImage() drawImage() method takes four parameters.

First parameter is the reference to the image that you want to display. second and third parameters indicate position of the image on Canvas.

Fourth parameter specifies the portion of the image bounding box that is placed at the specified co-ordinate position.

Example:

graphics.drawImage(img,5,20,Graphics.HCENTER | Graphics.VCENTER) palces the center of the image at co-ordinate position (5,20).

Wireless Markup Language (WML)

Introduction:

The Wireless Markup Language (WML) is the HTML of WAP browsers and is transmitted via the HTTP protocol.

WML is a markup language built specifically for communicating across WAP-based networks, and is based upon XML (eXtensible Markup Language). Like HDML, it is at first glance similar to HTML, but is also a much more strictly written language.

To make it possible for web pages to be read from a WAP-enabled device, WML must be used. The WML coder determines within the code what parts of the web page are viewable to the device, and what is not. For example, it would not be too advantageous for a 468x60 pixel banner to be loaded into the small screen of a WAP device, due to size, color and bandwidth restraints. However, certain parts of the text may be made available to the device.

The advantage of the WML language is the fact that, since it is a subset of XML, developer's can easily kill two birds with one stone by building both the web page and wireless device page simultaneously. While this is still possible with HDML code, it is certainly not as obvious and workarounds must be introduced

Why Should we use WML

Although you might not have any plans immediately for creating a WAP version of your site, it is always a good idea to get involved in new technology. All you need to do is make a small site (even one page) which tells people a bit about your website. In the future you can develop the site further with things like e-mail and information for people to get directly off their phones.

Over the past few months new WAP (Wireless Applications Protocol) phones have become extremely popular and many large websites have created special 'mobile' versions of their site. Many people predict that, over the next few years, WAP sites will become extremely popular and e-commerce over mobile phones will be widely available

The main sites which will benefit from WAP are ones providing a service like e-mail, live sports scores or a calendar service etc. but there are many other uses. For example, a site giving music reviews could put their reviews on a WAP site. People could then read the reviews on their mobile phone while browsing through the CDs in a shop.

WML Decks and Cards

WML pages are called DECKS. They are constructed as a set of CARDS, related to each other with links. When a WML page is accessed from a mobile phone, all the cards in the page are downloaded from the WAP server. Navigation between the cards is done by the phone computer - inside the phone - without any extra access trips to the server.

```
<?xml version="1.0"?>
!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.techiwarehouse.com/DTD/wml_1.1.xml">
<wml>
<card id="HTML" title="HTML Tutorial">
< p> You can learn WML in 30 Days
</p>
</card>
< card id="XML" title="XML Tutorial">
< p> You can also learn how to make an WAP based
Page < /p>
</card>
</wml>
```

As you can see from the example, the WML document is an XML document. The DOCTYPE is defined to be wml, and the DTD is accessed at www.techiwarehouse.org/DTD/wml_1.1.xml.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAP/DTD WML 1.1//EN"
"http://www.wap.org/DTD/wml_1.1.xml">
<wml>
<card id="no1" title="Card 1">
<p>Hello World!</p>
</card>
<card id="no2" title="Card 2">
<p>Welcome to our WAP Tutorial!</p>
</card>
</wml>
```

WML TAGS

WAP homepages are not very different from HTML homepages. The markup language used for WAP is WML (Wireless Markup Language). WML uses tags - just like HTML - but the syntax is stricter

and conforms to the XML 1.0 standard. WML pages have the extension *.WML, just like HTML pages have the extension *.HTML

WML is mostly about text. Tags that would slow down the communication with handheld devices are not a part of the WML standard. The use of tables and images is strongly restricted.

Since WML is an XML application, all tags are case sensitive (<wml> is not the same as </WML>), and all tags must be properly closed.

CONCLUSION

Wap's designed is flexible for client's minimal resources such as computing power and memory storage.WAP is programmed in wireless markup language WML (application of XML) and WMLScript (WAP's version JavaScript) which is embedded in client's mobile. WML provides a simple event mechanism that allows different content to be displayed. User actions, such as pressing a key, can be tied to scripts that cause changes in content. The WML browser also has this timer function that can load a different page or trigger the change of variables when the time is up. This provide great flexibility than the static content that HTML can deliver.

2.CALCULATOR

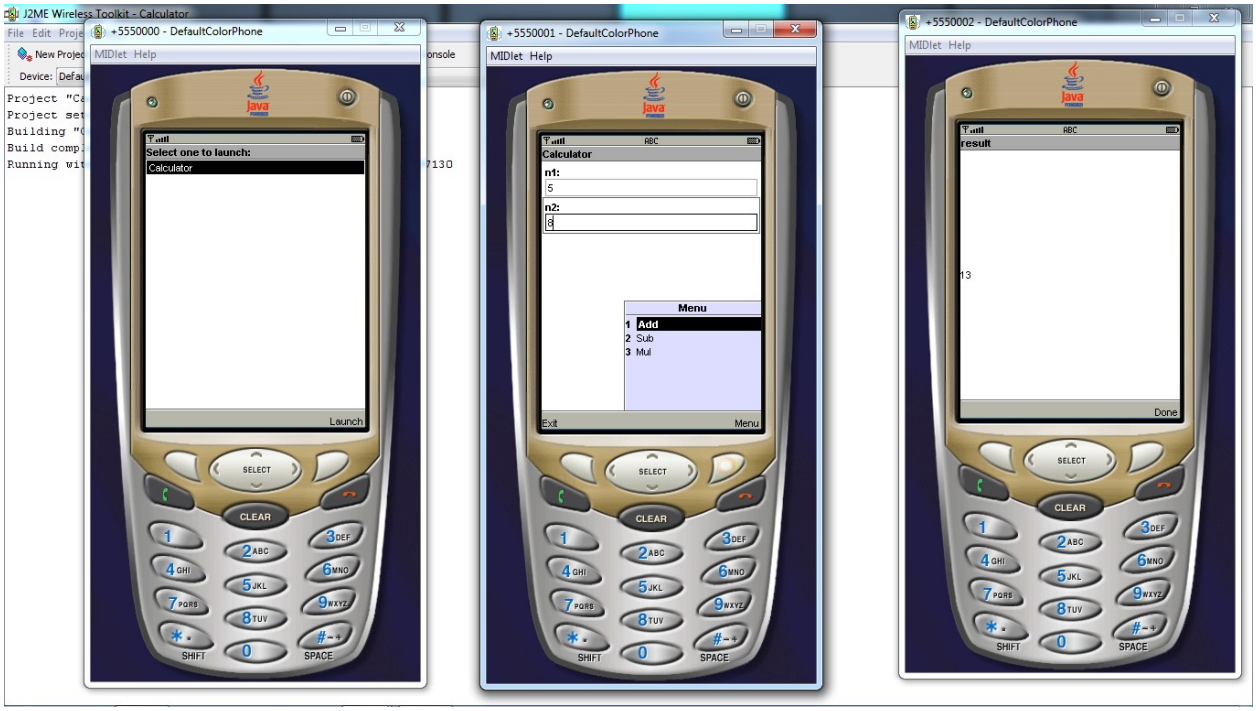
CODING:

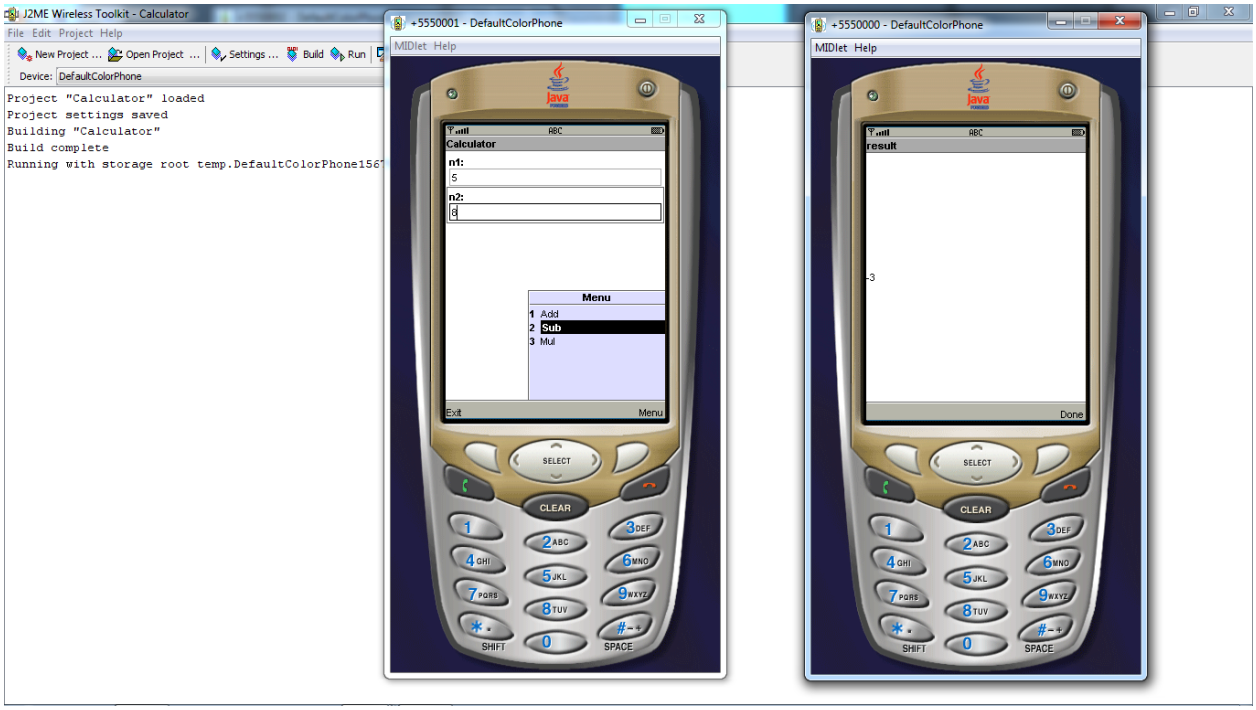
```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class Calculator extends MIDlet implements CommandListener
{
private Display display;
private Form form;
private TextField a,b,c;
private Command add,sub,mul,exit;
private Alert alert;
public Calculator()
{
a=new TextField("n1:","",30,TextField.ANY);
b=new TextField("n2:","",30,TextField.ANY);
add=new Command("Add",Command.OK,1);
sub=new Command("Sub",Command.OK,2);
mul=new Command("Mul",Command.OK,3);
exit=new Command("Exit",Command.EXIT,0);
display=Display.getDisplay(this);
}
public void startApp()
{
form=new Form("Calculator");
form.append(a);
form.append(b);
form.addCommand(add);
form.setCommandListener(this);
```

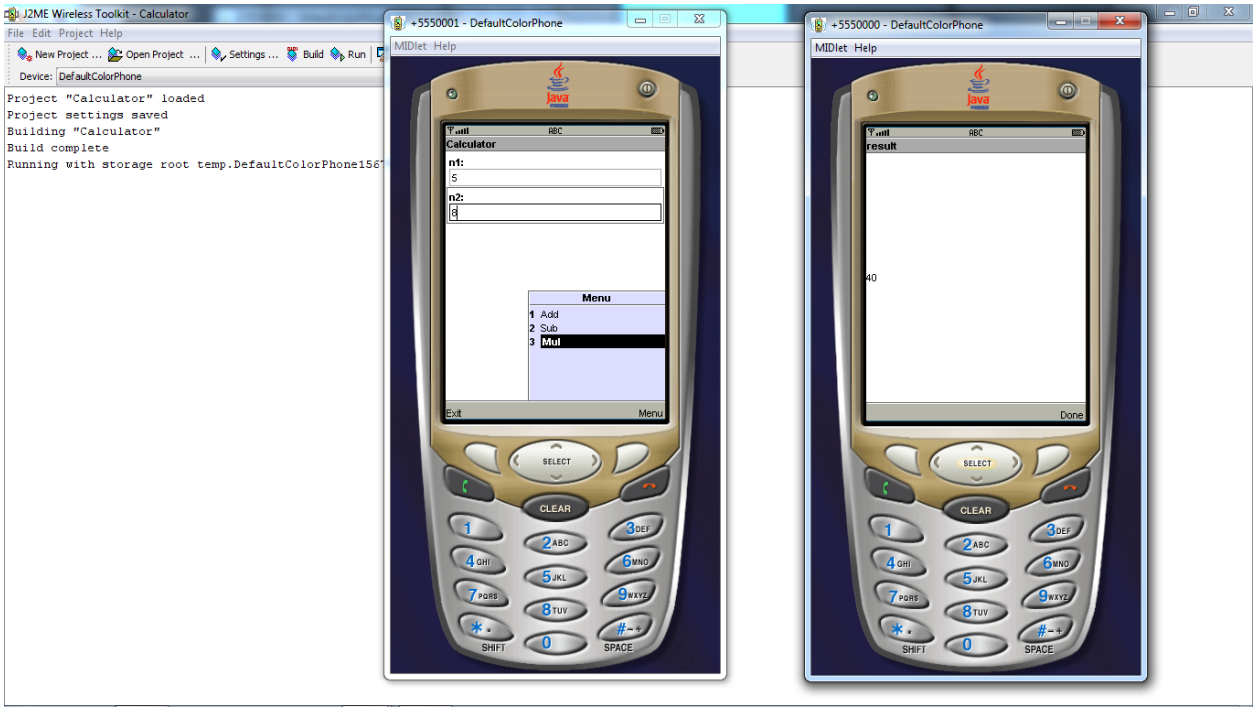
```
form.addCommand(sub);
form.setCommandListener(this);
form.addCommand(mul);
form.setCommandListener(this);
form.addCommand(exit);
form.setCommandListener(this);
display.setCurrent(form);
}
public void pauseApp()
{
}
public void destroyApp(boolean unconditional)
{
}
public void commandAction(Command c,Displayable s)
{
String n1=a.getString();
String n2=b.getString();
if(c==add)
{
display=Display.getDisplay(this);
int res=Integer.parseInt(n1)+Integer.parseInt(n2);
alert=new Alert("result",res+"",null,AlertType.INFO);
alert.setTimeout(Alert.FOREVER);
display.setCurrent(alert,form);
}
if(c==sub)
{
display=Display.getDisplay(this);
```

```
int res=Integer.parseInt(n1)-Integer.parseInt(n2);
alert=new Alert("result",res+"",null,AlertType.INFO);
alert.setTimeout(Alert.FOREVER);
display.setCurrent(alert,form);
}
if(c==mul)
{
display=Display.getDisplay(this);
int res=Integer.parseInt(n1)*Integer.parseInt(n2);
alert=new Alert("result",res+"",null,AlertType.INFO);
alert.setTimeout(Alert.FOREVER);
display.setCurrent(alert,form);
}
if(c==exit)
{
destroyApp(false);
notifyDestroyed();
} } }
```

OUTPUT:


J2ME Wireless Toolkit - Calculator
+5550000 - DefaultColorPhone
Select one to launch:
Calculator
Launch
+5550001 - DefaultColorPhone
Calculator
n1:
5
n2:
6
Menu
1 Add
2 Sub
3 Mul
Exit
Menu
+5550002 - DefaultColorPhone
result
13
Done


J2ME Wireless Toolkit - Calculator
New Project ... Open Project ... Settings ... Build Run
Device: DefaultColorPhone
Project "Calculator" loaded
Project settings saved
Building "Calculator"
Build complete
Running with storage root temp.DefaultColorPhone156
+5550001 - DefaultColorPhone
Calculator
n1:
5
n2:
6
Menu
1 Add
2 Sub
3 Mul
Exit
Menu
+5550000 - DefaultColorPhone
result
-3
Done


J2ME Wireless Toolkit - Calculator
File Edit Project Help
New Project ... Open Project ... Settings ... Build Run
Device: DefaultColorPhone
Project "Calculator" loaded
Project settings saved
Building "Calculator"
Build complete
Running with storage root temp.DefaultColorPhone156
+5550001 - DefaultColorPhone
MIDlet Help
Calculator
n1:
5
n2:
8
Menu
1 Add
2 Sub
3 Mul
Exit
Menu
SELECT
CLEAR
SHIFT
SPACE
+5550000 - DefaultColorPhone
MIDlet Help
result
40
Done

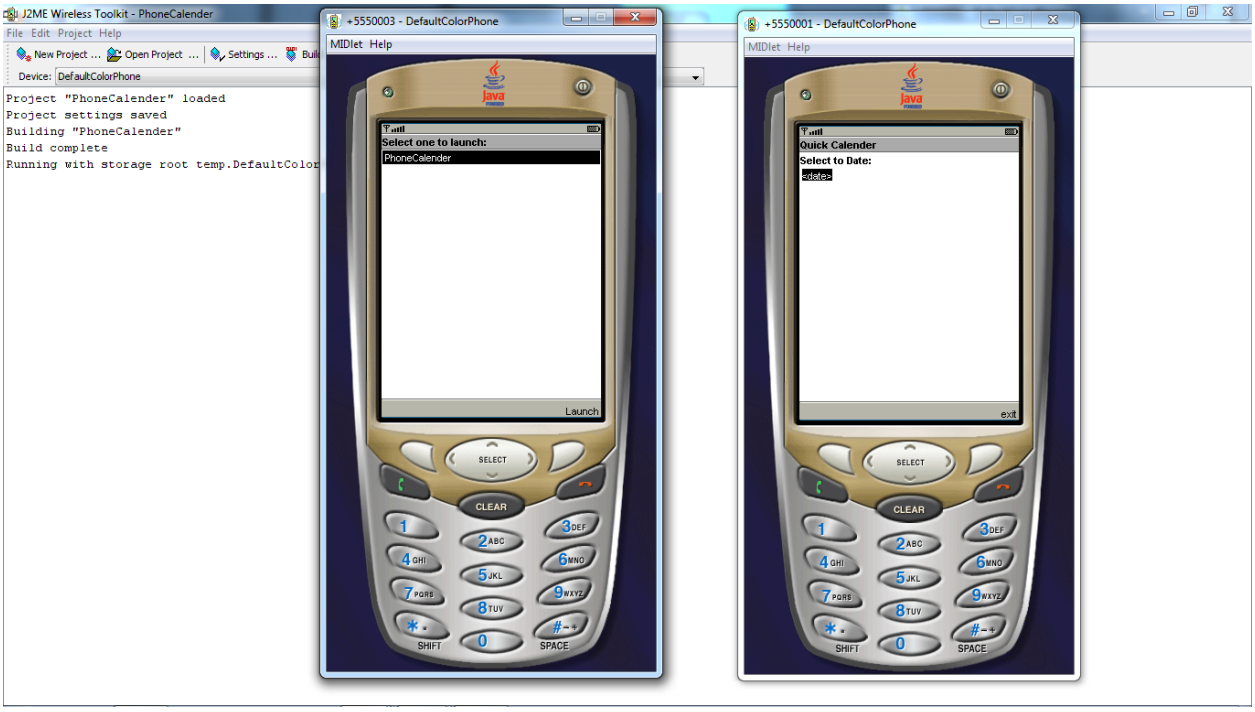
3.CALENDER

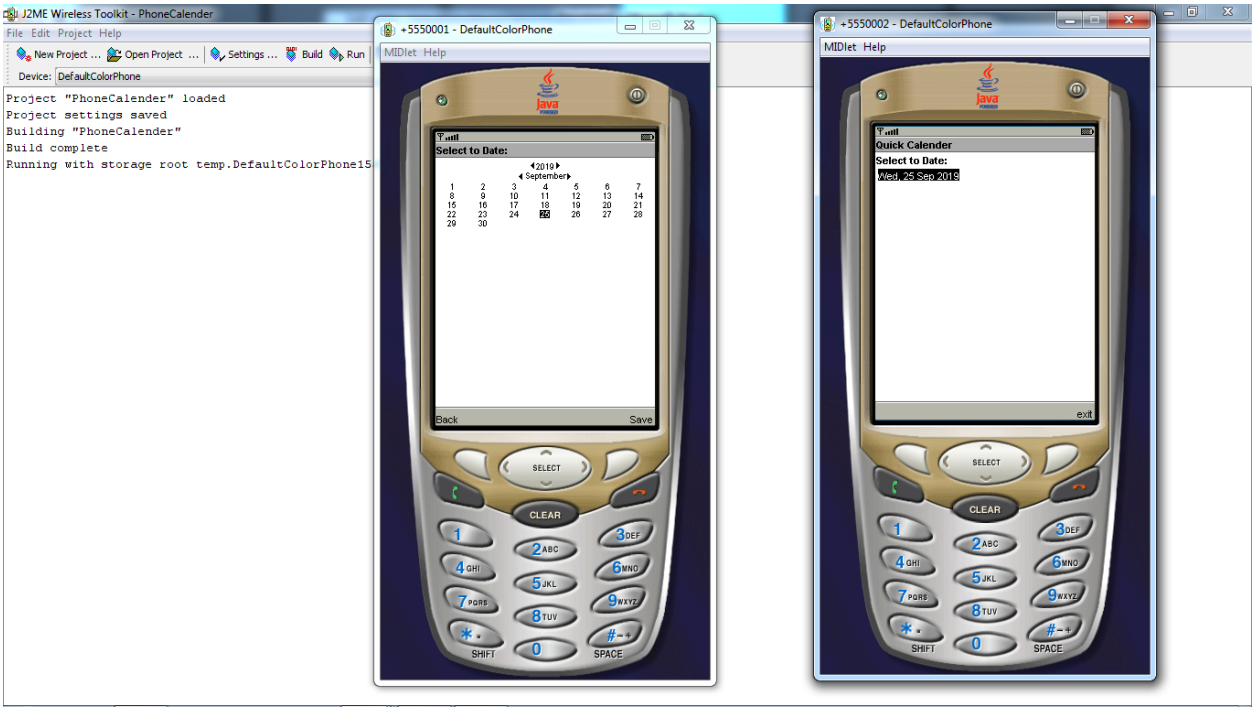
CODING:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class PhoneCalender extends MIDlet implements CommandListener,ItemStateListener
{
private Command exitCommand;
private Display display;
Form displayForm;
DateField date;
public PhoneCalender()
{
display=Display.getDisplay(this);
exitCommand=new Command("exit",Command.SCREEN,1);
date=new DateField("Select to Date:",DateField.DATE);
}
public void startApp()
{
displayForm=new Form("Quick Calender");
displayForm.append(date);
displayForm.addCommand(exitCommand);
displayForm.setCommandListener(this);
displayForm.setItemStateListener(this);
display.setCurrent(displayForm);
}
public void itemStateChanged(Item item)
{
}
public void pauseApp()
```

```
{}
public void destroyApp(boolean unconditional)
{}
public void commandAction(Command c,Displayable s)
{
if(c==exitCommand)
{
destroyApp(false);
notifyDestroyed();
}
}
}
```

OUTPUT:


J2ME Wireless Toolkit - PhoneCalender
File Edit Project Help
New Project ... Open Project ... Settings ... Bui
Device: DefaultColorPhone
Project "PhoneCalender" loaded
Project settings saved
Building "PhoneCalender"
Build complete
Running with storage root temp.DefaultColor
+5550003 - DefaultColorPhone
MIDlet Help
Select one to launch:
PhoneCalender
Launch
+5550001 - DefaultColorPhone
MIDlet Help
Quick Calender
Select to Date:
<date>
exit
SELECT
CLEAR
SHIFT
SPACE


J2ME Wireless Toolkit - PhoneCalender
File Edit Project Help
New Project ... Open Project ... Settings ... Build Run
Device: DefaultColorPhone
Project "PhoneCalender" loaded
Project settings saved
Building "PhoneCalender"
Build complete
Running with storage root temp.DefaultColorPhone15
+5550001 - DefaultColorPhone
MIDlet Help
Select to Date:
2019
September
Back
Save
+5550002 - DefaultColorPhone
MIDlet Help
Quick Calender
Select to Date:
Wed, 25 Sep 2019
exit
SELECT
CLEAR
SHIFT
SPACE

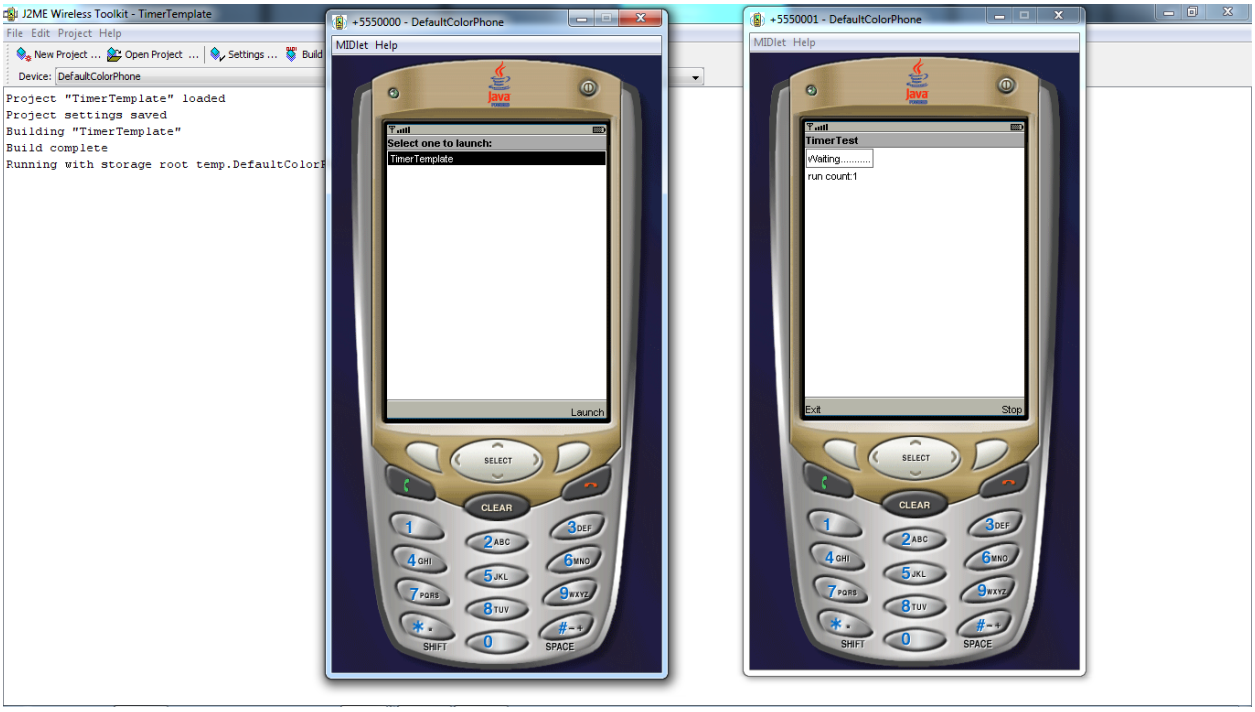
4.TIMERTEMPLATE

CODING:

```
import java.util.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class TimerTemplate extends MIDlet implements CommandListener
{
private Display display;
private Form fmMain;
private Command cmExit;
private Command cmStop;
private Timer tm;
private TestTimerTask tt;
private int count=0;
public TimerTemplate()
{
display=Display.getDisplay(this);
fmMain=new Form("TimerTest");
fmMain.append("Waiting.....\n");
cmExit=new Command("Exit",Command.EXIT,1);
cmStop=new Command("Stop",Command.STOP,2);
fmMain.addCommand(cmExit);
fmMain.addCommand(cmStop);
fmMain.setCommandListener(this);
tm=new Timer();
tt=new TestTimerTask();
tm.schedule(tt,5000);
}
public void startApp()
```

```
{
display.setCurrent(fmMain);
}
public void destroyApp(boolean unconditional)
{
}
public void pauseApp()
{
}
public void commandAction(Command c,Displayable d)
{
if(c==cmStop)
{
tm.cancel();
}
else if(c==cmExit)
{
destroyApp(false);
notifyDestroyed();
}
}
private class TestTimerTask extends TimerTask
{
public final void run()
{
fmMain.append("run count:"+ ++count+);
}
}
}
```

OUTPUT:


J2ME Wireless Toolkit - TimerTemplate
File Edit Project Help
New Project ... Open Project ... Settings ... Build
Device: DefaultColorPhone
Project "TimerTemplate" loaded
Project settings saved
Building "TimerTemplate"
Build complete
Running with storage root temp.DefaultColor
+5550000 - DefaultColorPhone
MIDlet Help
Select one to launch:
TimerTemplate
Launch
+5550001 - DefaultColorPhone
MIDlet Help
TimerTest
Waiting.....
run count:1
Exit
Stop
SELECT
CLEAR
SHIFT
SPACE

5.SIMPLE GAME

CODING:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.Timer;
import java.util.TimerTask;
public class Bouncing extends MIDlet
{
Ball b1,b2;
BallCanvas bc;
Display display;
public Bouncing()
{
System.out.println("Bouncing ball");
}
public void destroyApp(boolean a)
{
 }
public void startApp()
{
try
{
bc=new BallCanvas();
display=Display.getDisplay(this);
Ball b1=new Ball(90,30,5,5,1,10,250,255,255,bc.getGraphics(),bc.getWidth(),bc.getHeight(),bc);
Ball b2=new Ball(100,20,15,1,3,20,150,150,250,bc.getGraphics(),bc.getWidth(),bc.getHeight(),bc);
Ball b3=new Ball(10,10,30,2,1,25,250,100,250,bc.getGraphics(),bc.getWidth(),bc.getHeight(),bc);
display.setCurrent(bc);
```

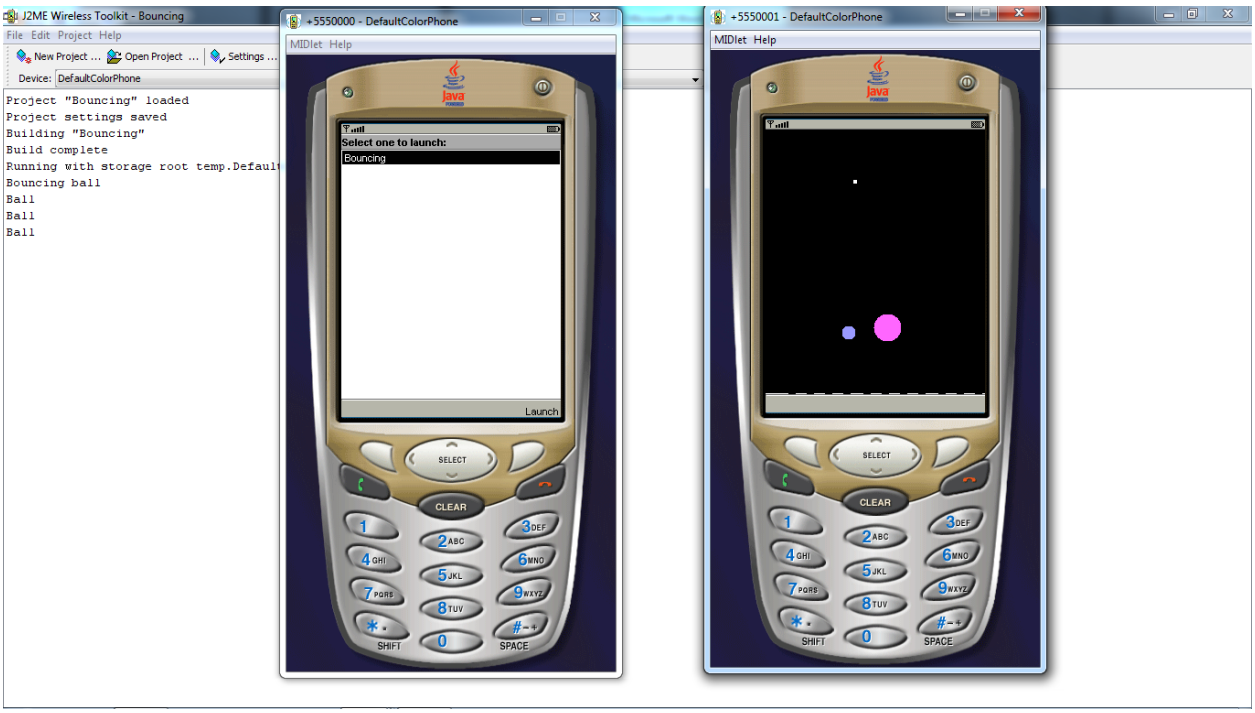
```
}
catch(Exception e)
{
System.out.println(e);
}
}
public void pauseApp()
{
}
public class BallCanvas extends Canvas
{
Graphics g1;
public void paint(Graphics g)
{
g1=g;
}
Graphics getGraphics()
{
return(g1);
}
}
public class Ball
{
private int ox,oy,redius,speed;
private int incx,incy;
private int red,green,blue;
private int maxx,maxy;
private boolean xb,yb,flag;
TimerTask t;
```

```
Timer timer;
Graphics g;
BallCanvas bc;
private int prev1,prev2;
public Ball(int x,int y,int r,int ix,int iy,int s,int rd,int gr,int b,Graphics g1,int mx,int my,BallCanvas bc1)
{
System.out.println("Ball");
maxx=mx;
maxy=my;
incx=ix;
incy=iy;
red=rd;
green=gr;
blue=b;
xb=false;
yb=false;
redius=r;
ox=x;
oy=y;
flag=false;
bc=bc1;
g1=g;
timer=new Timer();
timer.schedule(new Bounce(),(long)0,(long)s);
}
public void paint()
{
if((ox+redius)>=maxx)
xb=true;
```

```
if(ox<=0)
xb=false;
if((oy+redius)>=maxy)
yb=true;
if(oy<=0)
yb=false;
if(xb==false)
{
prev1=ox;
ox+=incx;
}
else
{
prev1=ox;
ox-=incx;
}
if(yb==false)
{
prev2=oy;
oy+=incy;
}
else
{
prev2=oy;
oy-=incy;
}
bc.repaint();
if(flag==false)
{
```

```
bc.getGraphics().setColor(0,0,0);
bc.getGraphics().fillRect(0,0,bc.getWidth(),bc.getHeight());
flag=true;
}
bc.getGraphics().setColor(0,0,0);
bc.getGraphics().fillArc(prev1,prev2,redius,redius,0,360);
bc.getGraphics().setColor(red,green,blue);
bc.getGraphics().fillArc(ox,oy,redius,redius,0,360);
try
{
wait(20);
}
catch(Exception e)
{
}
}
public class Bounce extends TimerTask
{
public void run()
{
paint();
}
}
}
}
```

OUTPUT:


J2ME Wireless Toolkit - Bouncing
File Edit Project Help
New Project ... Open Project ... Settings ...
Device: DefaultColorPhone
Project "Bouncing" loaded
Project settings saved
Building "Bouncing"
Build complete
Running with storage root temp.Default
Bouncing ball
Ball
Ball
Ball
+5550000 - DefaultColorPhone
MIDlet Help
Select one to launch:
Bouncing
Launch
SELECT
CLEAR
SHIFT
SPACE
+5550001 - DefaultColorPhone
MIDlet Help
SELECT
CLEAR
SHIFT
SPACE

6.ANIMATION

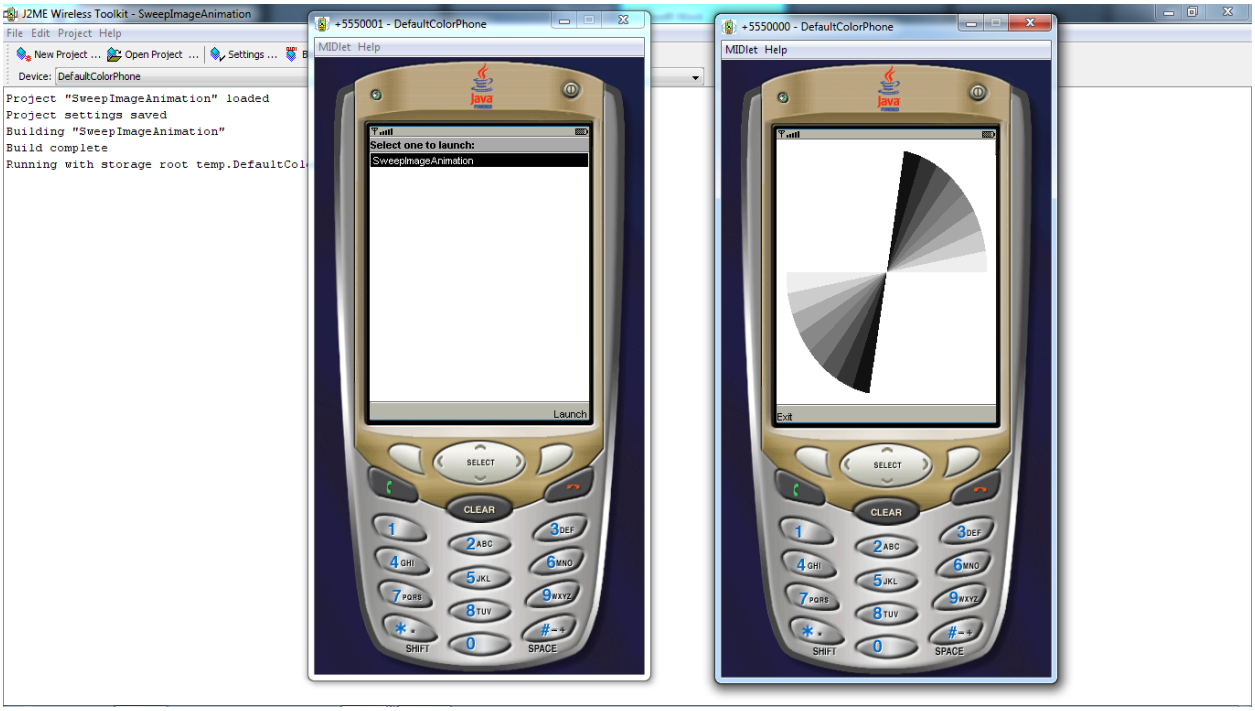
CODING:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class SweepImageAnimation extends MIDlet
{
public void startApp()
{
final SweepCanvas sweeper=new SweepCanvas();
sweeper.start();
sweeper.addCommand(new Command("Exit",Command.EXIT,0));
sweeper.setCommandListener(new CommandListener()
{
public void commandAction(Command c,Displayable s)
{
sweeper.stop();
notifyDestroyed();
}
});
Display.getDisplay(this).setCurrent(sweeper);
}
public void pauseApp()
{
}
public void destroyApp(boolean unconditional)
{
}
}
```

```
class SweepCanvas extends Canvas implements Runnable
{
private boolean mTrucking;
private int mTheta;
private int mBorder;
private int mDelay;
public SweepCanvas()
{
mTheta=0;
mBorder=10;
mDelay=50;
}
public void start()
{
mTrucking=true;
Thread t=new Thread(this);
t.start();
}
public void stop()
{
mTrucking=false;
}
public void paint(Graphics g)
{
int width=getWidth();
int height=getHeight();
//Clear the Canvas
g.setGrayScale(255);
g.fillRect(0,0,width-1,height-1);
```

```
int x=mBorder;
int y=mBorder;
int w=width-mBorder*2;
int h=height-mBorder*2;
for(int i=0;i<8;i++)
{
g.setGrayScale((8-i)*32-16);
g.fillArc(x,y,w,h,mTheta+i*10,10);
g.fillArc(x,y,w,h,(mTheta+180)%360+i*10,10);
}
}
public void run()
{
while(mTrucking)
{
mTheta=(mTheta+1)%360;
repaint();
try
{
Thread.sleep(mDelay);
}
catch(InterruptedException ie)
{
}
}
}
}
```

OUTPUT:


J2ME Wireless Toolkit - SweepImageAnimation
File Edit Project Help
New Project ... Open Project ... Settings ...
Device: DefaultColorPhone
Project "SweepImageAnimation" loaded
Project settings saved
Building "SweepImageAnimation"
Build complete
Running with storage root temp.DefaultCol
+5550001 - DefaultColorPhone
MIDlet Help
Select one to launch:
SweepImageAnimation
Launch
+5550000 - DefaultColorPhone
MIDlet Help
Exit
SELECT
CLEAR
SHIFT
SPACE

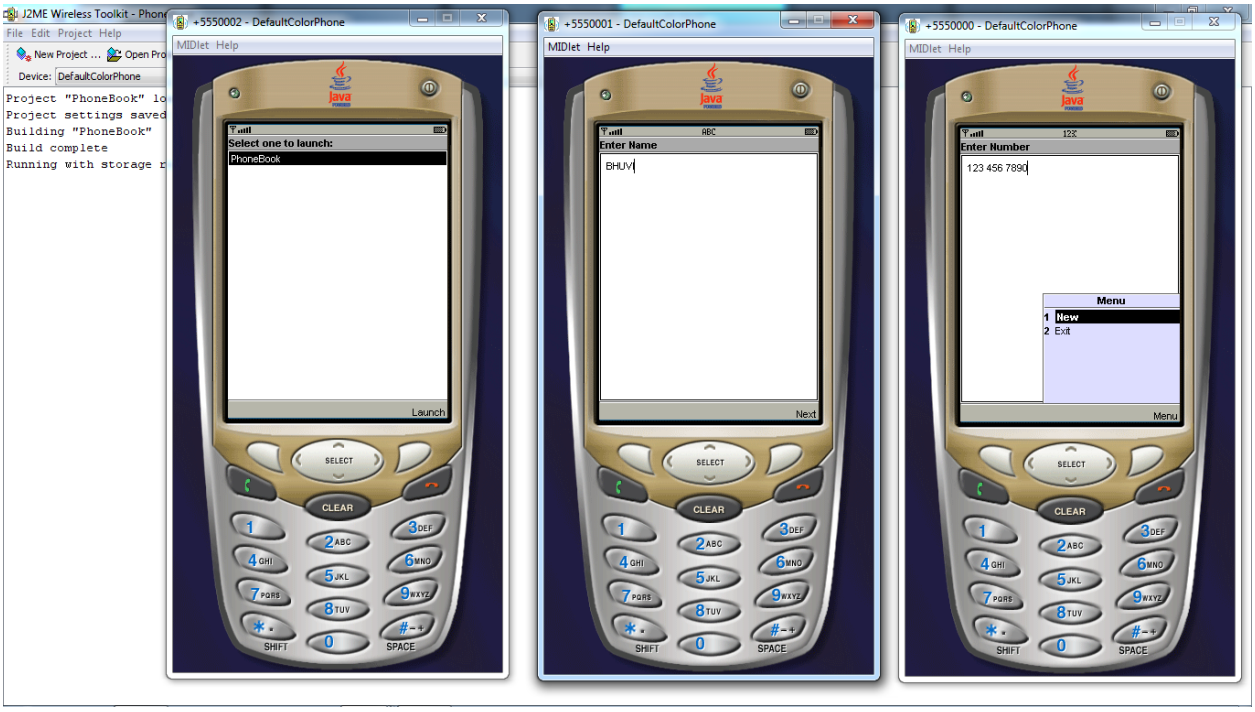
7.PERSONAL PHONE BOOK

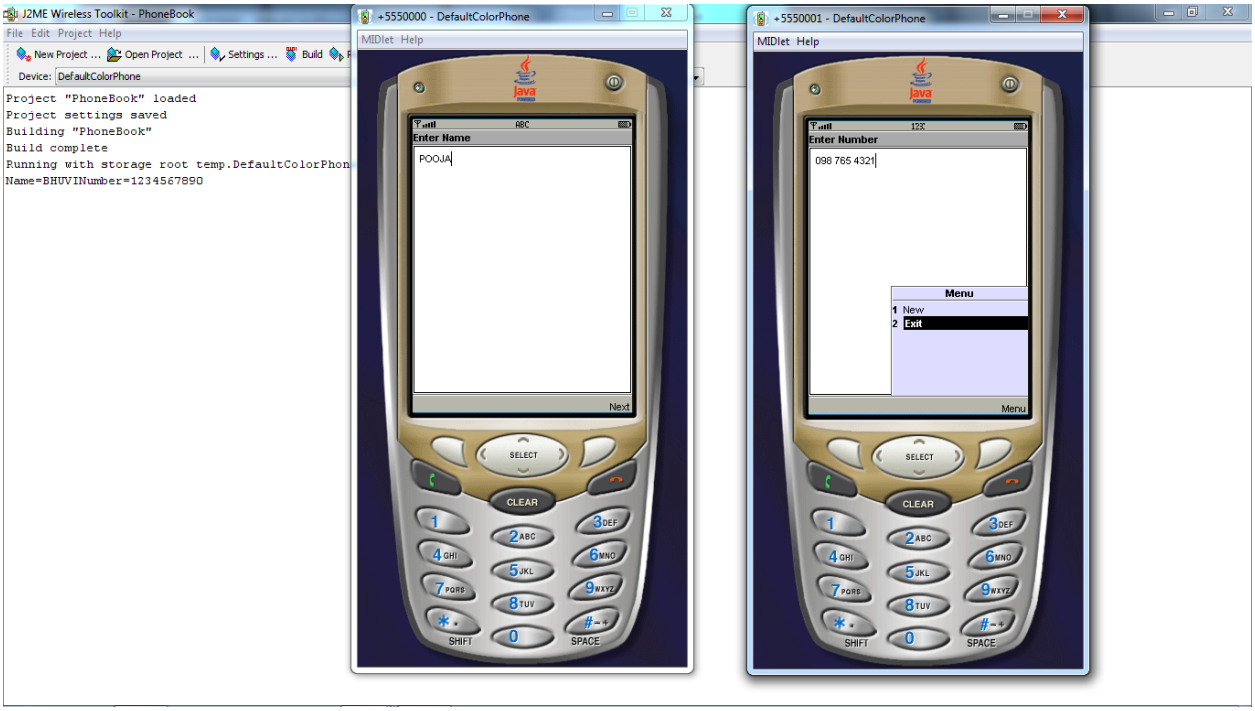
CODING:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class PhoneBook extends MIDlet implements CommandListener
{
private Command exit,next,New;
private TextBox name,number;
private Display display;
public PhoneBook()
{
next=new Command("Next",Command.SCREEN,2);
exit=new Command("Exit",Command.SCREEN,2);
New=new Command("New",Command.SCREEN,2);
name=new TextBox("Enter Name","",30,TextField.ANY);
number=new TextBox("Enter Number","",13,TextField.PHONENUMBER);
}
public void startApp()
{
display=Display.getDisplay(this);
name.addCommand(next);
name.setCommandListener(this);
number.addCommand(New);
number.setCommandListener(this);
number.addCommand(exit);
number.setCommandListener(this);
display.setCurrent(name);
}
public void pauseApp()
```

```
{
}
public void destroyApp(boolean unconditional)
{
notifyDestroyed();
}
public void commandAction(Command c,Displayable s)
{
String label=c.getLabel();
if(label.equals("Exit"))
{
System.out.println("Name="+name.getString()+"Number="+number.getString());
destroyApp(false);
}
else if(label.equals("Next"))
{
number.setString("");
display.setCurrent(number);
}
else if(label.equals("New"))
{
display.setCurrent(name);
System.out.println("Name="+name.getString()+"Number="+number.getString());
name.setString("");
}
}
}
```

OUTPUT:


J2ME Wireless Toolkit - Phone
File Edit Project Help
New Project ... Open Pro
Device: DefaultColorPhone
Project "PhoneBook" lo
Project settings saved
Building "PhoneBook"
Build complete
Running with storage r
+5550002 - DefaultColorPhone
MIDlet Help
Select one to launch:
PhoneBook
Launch
+5550001 - DefaultColorPhone
MIDlet Help
Enter Name
BHUV
Next
+5550000 - DefaultColorPhone
MIDlet Help
Enter Number
123 456 7890
Menu
1 New
2 Exit
Menu
SELECT
CLEAR
SHIFT
SPACE


J2ME Wireless Toolkit - PhoneBook
File Edit Project Help
New Project ... Open Project ... Settings ... Build
Device: DefaultColorPhone
Project "PhoneBook" loaded
Project settings saved
Building "PhoneBook"
Build complete
Running with storage root temp.DefaultColorPhon
Name=BHUVINumber=1234567890
+5550000 - DefaultColorPhone
MIDlet Help
Enter Name
POOJA
Next
+5550001 - DefaultColorPhone
MIDlet Help
Enter Number
098 765 4321
Menu
1 New
2 Exit
Menu
SELECT
CLEAR
SHIFT
SPACE

8.AUTHENTICATION AND ENCRYPTION TECHNIQUE

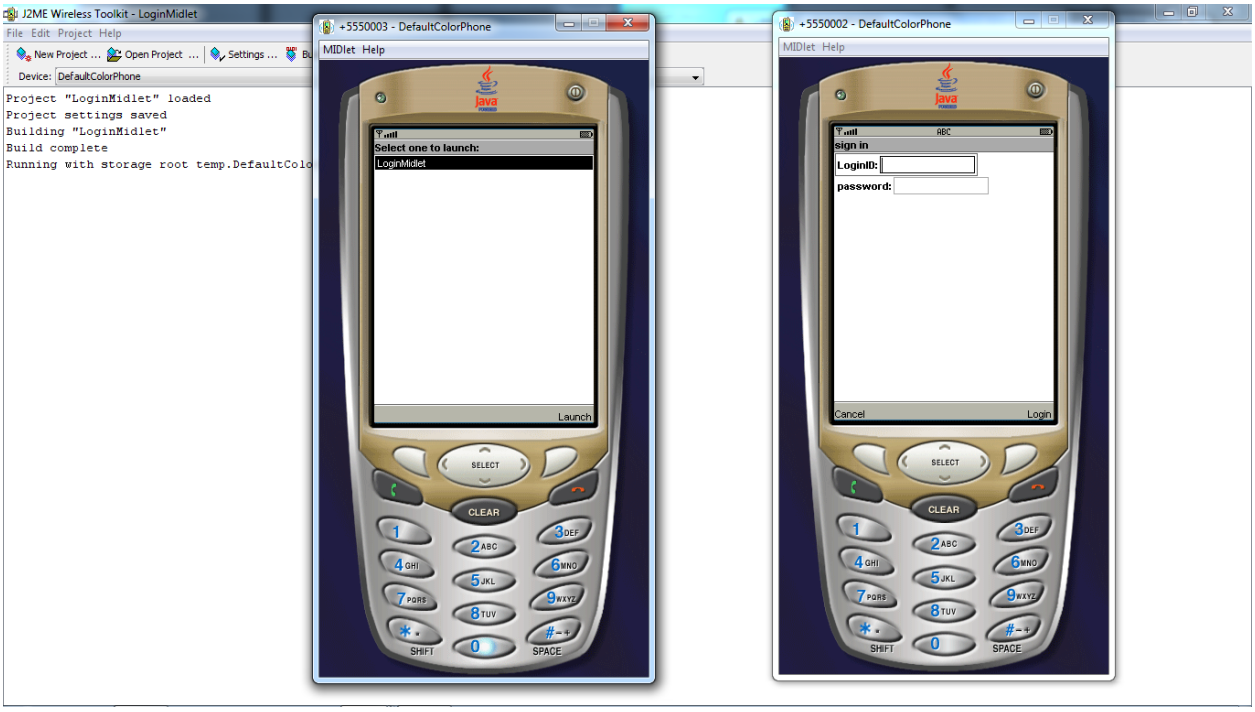
CODING:

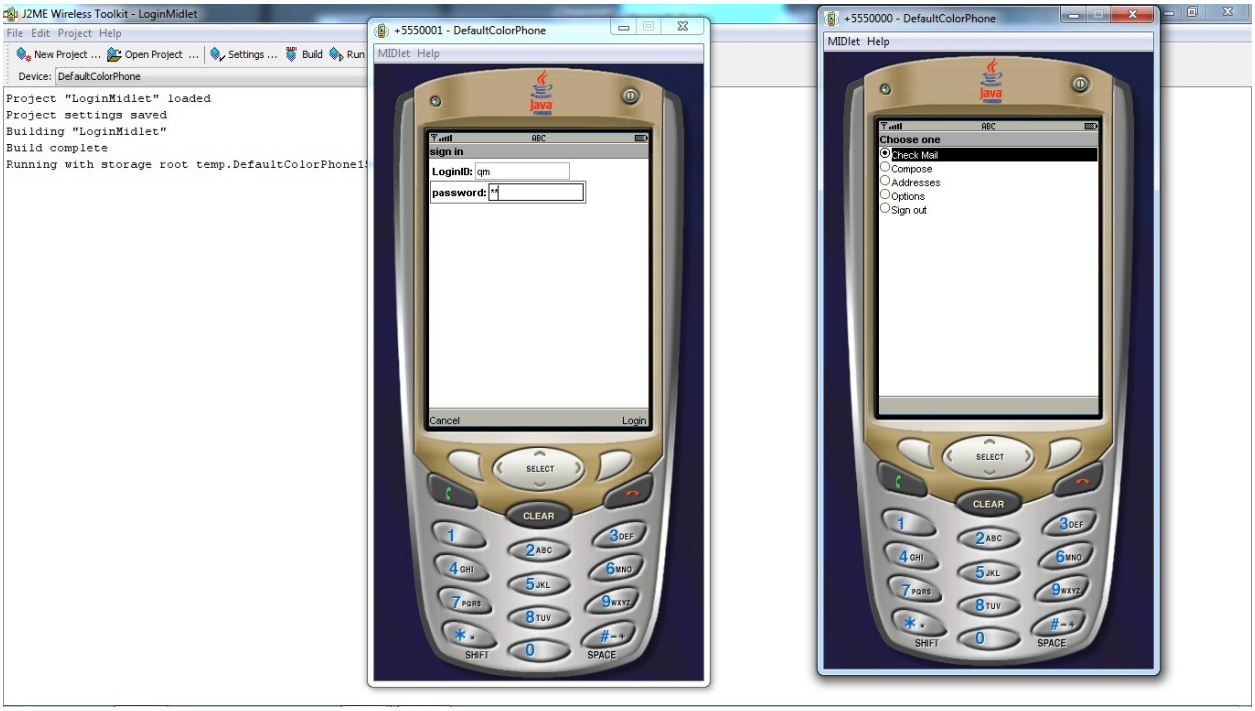
```
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.*;
public class LoginMidlet extends MIDlet implements CommandListener
{
private Display display;
private TextField userName;
private TextField password;
private Form form;
private Command cancel;
private Command login;
public LoginMidlet()
{
userName=new TextField("LoginID:","",10,TextField.ANY);
password=new TextField("password:","",10,TextField.PASSWORD);
form=new Form("sign in");
cancel=new Command("Cancel",Command.CANCEL,2);
login=new Command("Login",Command.OK,2);
}
public void startApp()
{
display=Display.getDisplay(this);
form.append(userName);
form.append(password);
form.addCommand(cancel);
form.setCommandListener(this);
form.addCommand(login);
form.setCommandListener(this);
```

```
display.setCurrent(form);
}
public void pauseApp()
{ }
public void destroyApp(boolean unconditional)
{
notifyDestroyed();
}
public void validateUser(String name,String password)
{
if(name.equals("qm")&&password.equals("j2"))
{
menu();
}
else
{
tryAgain();
} }
public void menu()
{
List services=new List("Choose one",Choice.EXCLUSIVE);
services.append("Check Mail",null);
services.append("Compose",null);
services.append("Addresses",null);
services.append("Options",null);
services.append("Sign out",null);
display.setCurrent(services);
}
public void tryAgain() {
```

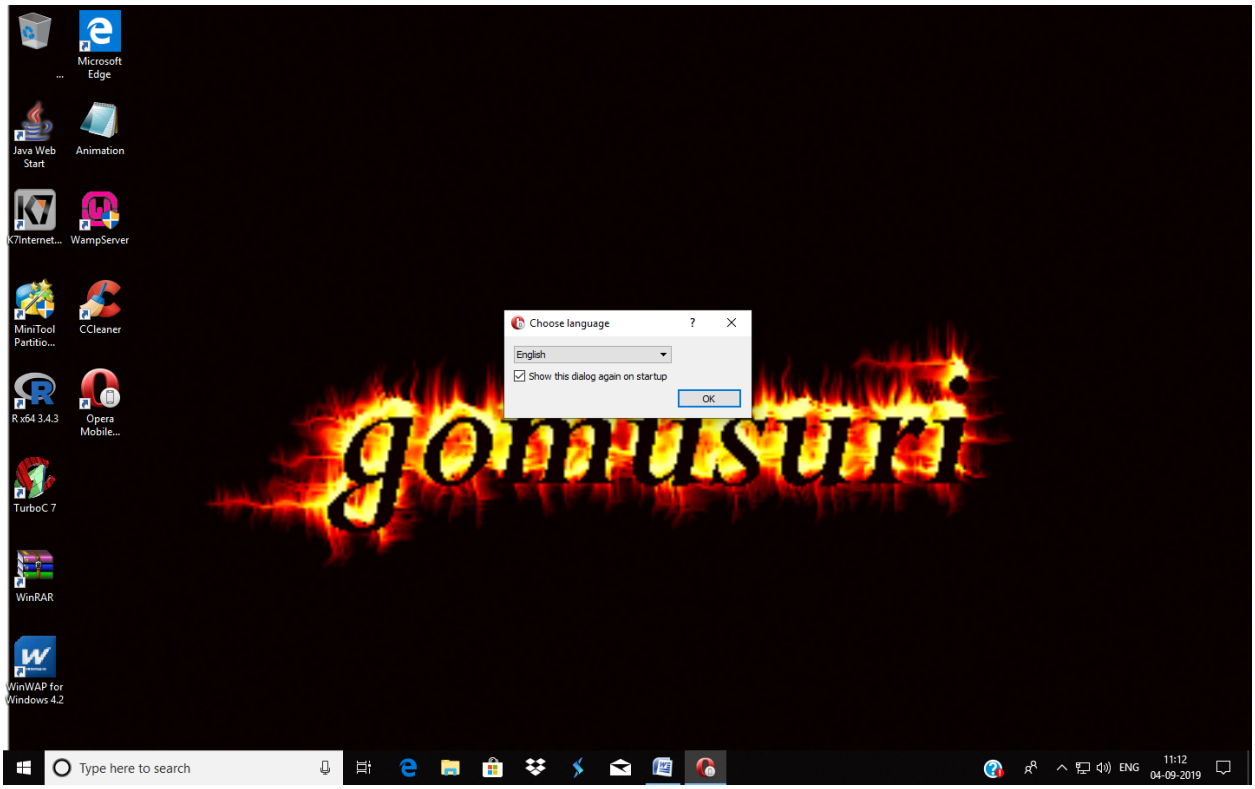
```
Alert error=new Alert("Login Incorrect","Please try again",null,AlertType.ERROR);
error.setTimeout(Alert.FOREVER);
userName.setString("");
password.setString("");
display.setCurrent(error,form);
}
public void commandAction(Command c,Displayable d)
{
String label=c.getLabel();
if(label.equals("Cancel")) {
destroyApp(true);
}
else if(label.equals("Login"))
{
validateUser(userName.getString(),password.getString());
}
} }
```

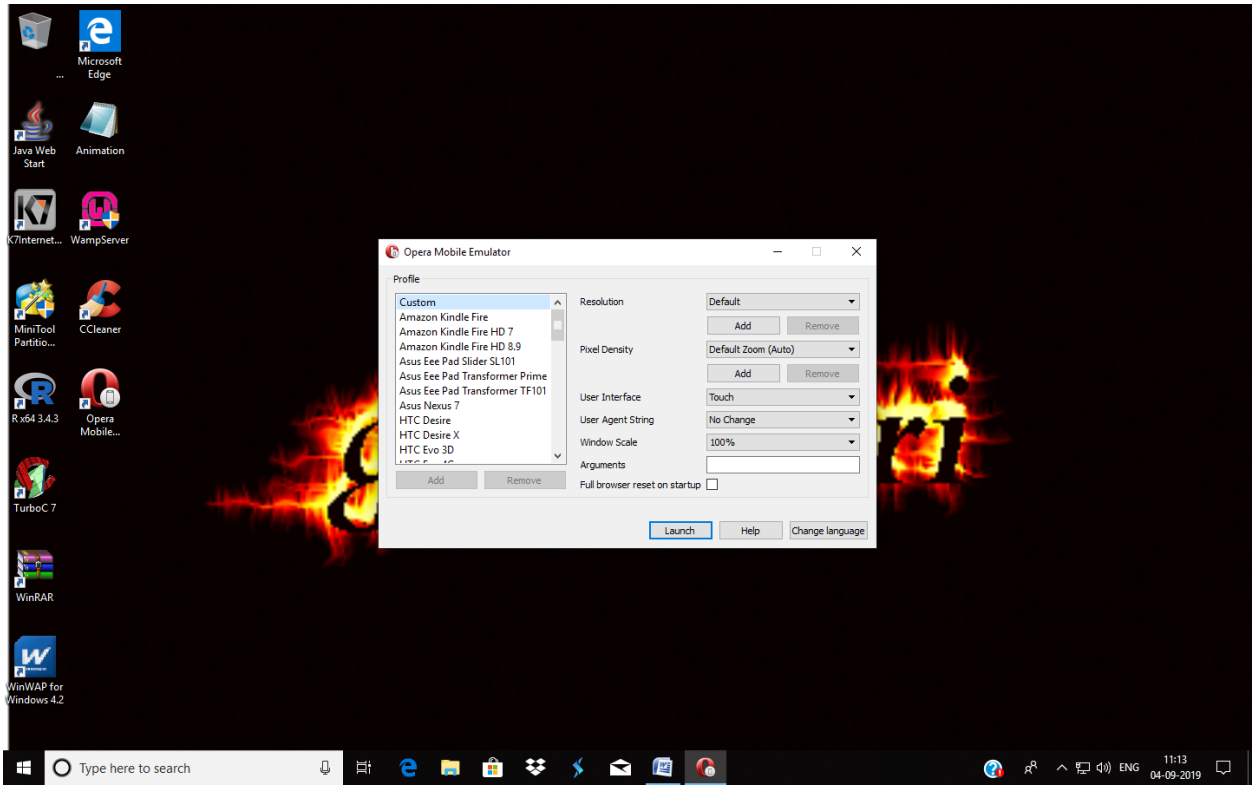
OUTPUT:

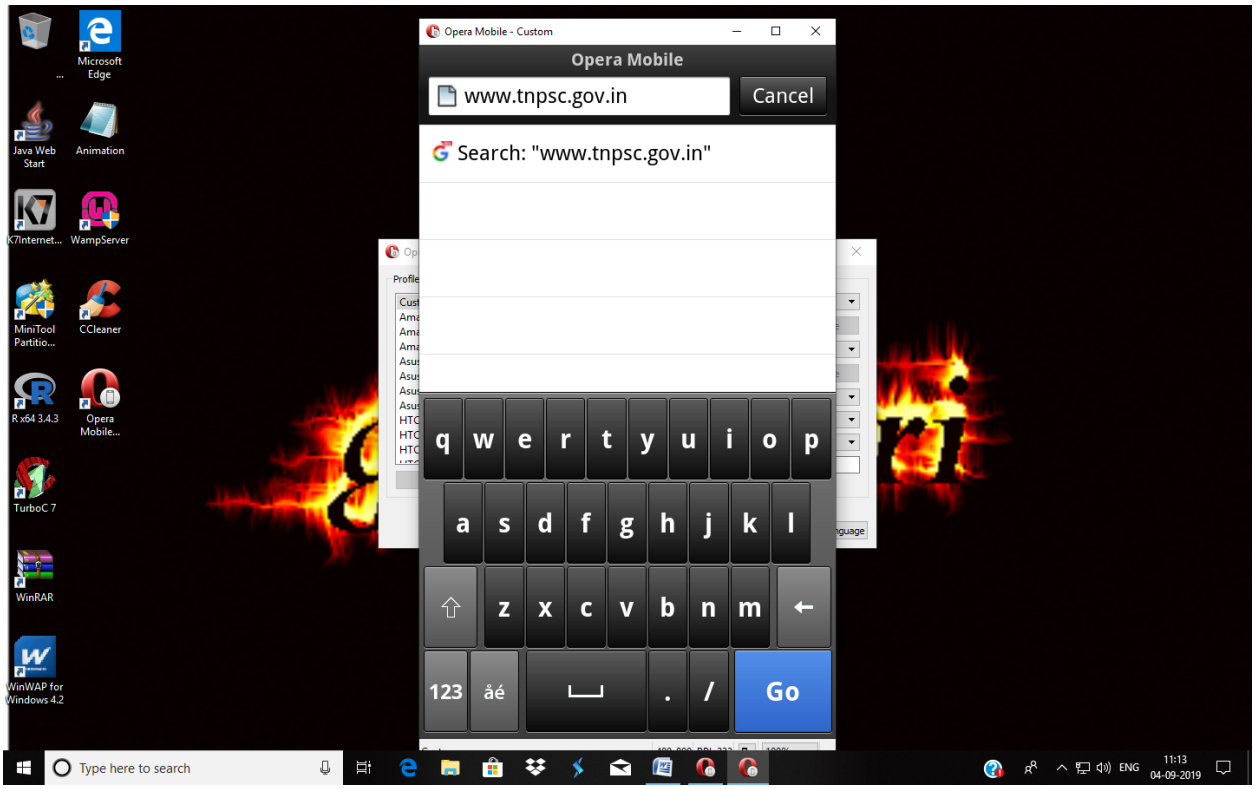

J2ME Wireless Toolkit - LoginMidlet
File Edit Project Help
New Project ... Open Project ... Settings ...
Device: DefaultColorPhone
Project "LoginMidlet" loaded
Project settings saved
Building "LoginMidlet"
Build complete
Running with storage root temp.DefaultColo
+5550003 - DefaultColorPhone
MIDlet Help
Select one to launch:
LoginMidlet
Launch
+5550002 - DefaultColorPhone
MIDlet Help
sign in
LoginID:
password:
Cancel
Login
SELECT
CLEAR
SHIFT
SPACE

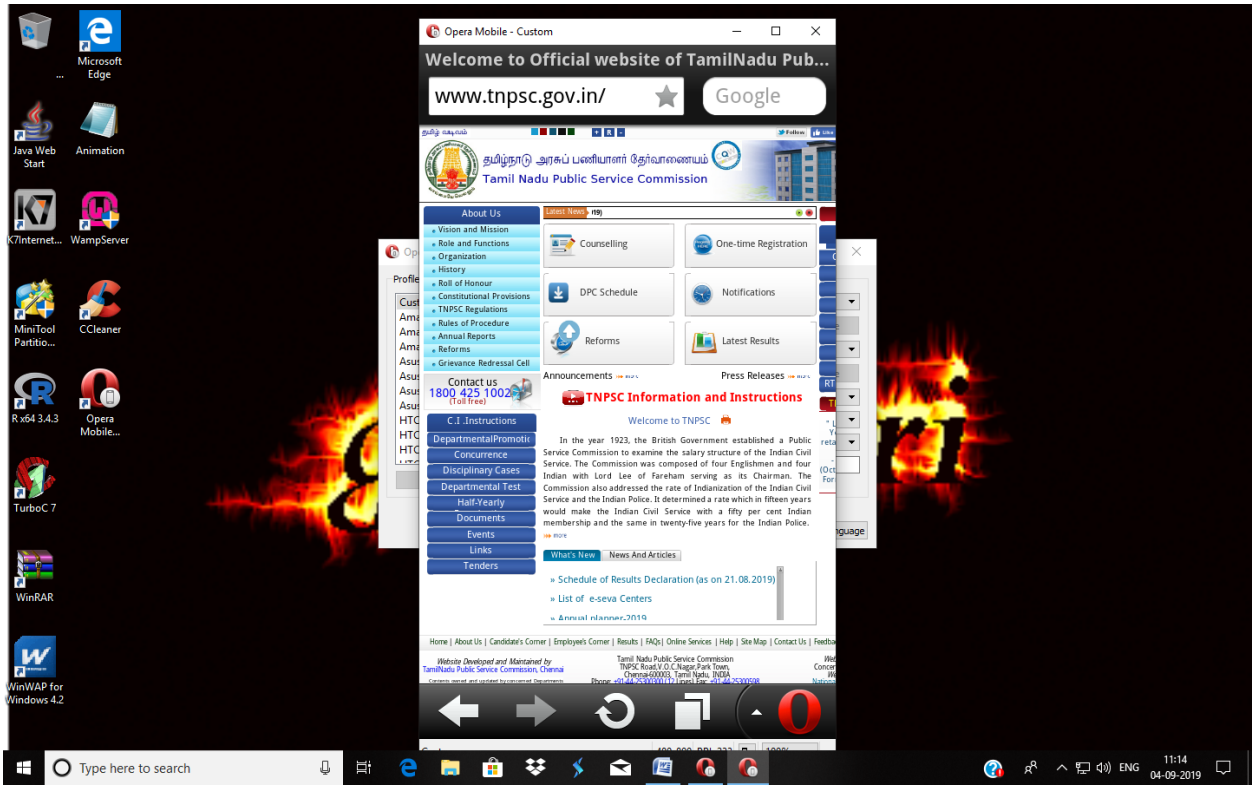

J2ME Wireless Toolkit - LoginMidlet
File Edit Project Help
New Project ... Open Project ... Settings ... Build Run
Device: DefaultColorPhone
Project "LoginMidlet" loaded
Project settings saved
Building "LoginMidlet"
Build complete
Running with storage root temp.DefaultColorPhone1
+5550001 - DefaultColorPhone
MIDlet Help
sign in
LoginID: qm
password: **
Cancel
Login
+5550000 - DefaultColorPhone
MIDlet Help
Choose one
Check Mail
Compose
Addresses
Options
Sign out
SELECT
CLEAR
SHIFT
SPACE

9.SIMULATION


Microsoft Edge
Java Web Start
Animation
K7Internet...
WampServer
MiniTool Partitio...
CCleaner
R x64 3.4.3
Opera Mobile...
TurboC 7
WinRAR
WinWAP for Windows 4.2
Choose language
English
Show this dialog again on startup
OK
gomusuri
Type here to search
ENG
11:12
04-09-2019


Microsoft Edge
Java Web Start
Animation
K7Internet...
WampServer
MiniTool Partitio...
CCleaner
R x64 3.4.3
Opera Mobile...
TurboC 7
WinRAR
WinWAP for Windows 4.2
Opera Mobile Emulator
Profile
Custom
Amazon Kindle Fire
Amazon Kindle Fire HD 7
Amazon Kindle Fire HD 8.9
Asus Eee Pad Slider SL101
Asus Eee Pad Transformer Prime
Asus Eee Pad Transformer TF101
Asus Nexus 7
HTC Desire
HTC Desire X
HTC Evo 3D
Add
Remove
Resolution
Default
Add
Remove
Pixel Density
Default Zoom (Auto)
Add
Remove
User Interface
Touch
User Agent String
No Change
Window Scale
100%
Arguments
Full browser reset on startup
Launch
Help
Change language
Type here to search
ENG
11:13
04-09-2019


Opera Mobile - Custom
Opera Mobile
www.tnpsc.gov.in
Cancel
Search: "www.tnpsc.gov.in"
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123 åé . / Go
Type here to search
ENG
11:13
04-09-2019


Opera Mobile - Custom
Welcome to Official website of TamilNadu Pub...
www.tnpsc.gov.in/
Google
தமிழ்நாடு அரசுப் பணியாளர் தேர்வாணையம்
Tamil Nadu Public Service Commission
About Us
Counselling
One-time Registration
DPC Schedule
Notifications
Reforms
Latest Results
Contact us
1800 425 1002
TNPSC Information and Instructions
Welcome to TNPSC
Type here to search
ENG
11:14
04-09-2019

10.Study of GlomoSim Simulator

Aim:

To study the concept of Glomo sim Simulator.

1. Introduction to GloMoSim.

1. Introduction

With GloMoSim we are building a scalable simulation environment for wireless network systems. It is being designed using the parallel discrete-event simulation capability provided by Parsec.

Most network systems are currently built using a layered approach that is similar to the OSI seven layer network architecture. The plan is to build GloMoSim using a similar layered approach. Standard APIs will be used between the different simulation layers. This will allow the rapid integration of models developed at different layers by different people. The goal is to build a library of parallelized models that can be used for the evaluation of a variety of wireless network protocols. The proposed protocol stack will include models for the channel, radio, MAC, network, transport, and higher layers.

1.1 Network Gridding

The simple approach to designing a network simulation would be to initialize each network node in the simulation as a Parsec entity. We can view different entity initializations as being separate logical processes in the system. Hence each entity initialization requires its own stack space in the runtime. In GloMoSim, we are trying to build a simulation that will scale to thousands of nodes. If we have to instantiate an entity for each node in the runtime, the memory requirements would increase dramatically. The performance of the system would also degrade rapidly. Since there are so many entities in the simulation, the runtime would need to constantly context switch among the different entities in the system. This will cause significant degradation in the performance of the simulation. Hence initializing each node as a separate entity will inherently limit the scalability and performance of the simulation.

To circumvent these problems network gridding was introduced into the simulation. With network gridding, a single entity can simulate several network nodes in the system. A separate data structure representing the complete state of each node is maintained within the entity. Similarly we need to maintain the right level of abstraction. When the simulation code of a particular node is being executed it should not have access to the data structures of the other nodes in the simulation. The network gridding technique means that we can increase the number of nodes in the system while maintaining the same number of entities in the simulation. In fact, the only requirement is that we need only as many entities as the number of processors on which the simulation is being run. Hence if we are running a sequential simulation we need to initialize only one entity in the system. We also don't meet the memory or context switching problems that limit the simulation.

In GloMoSim, each entity represents a geographical area of the simulation. Hence the network nodes which a particular entity represents are determined by the physical position of the nodes.

Suppose we specify the following in the input file:

```
#
SIMULATION-RANGE-X 100
SIMULATION-RANGE-Y 100
#
# Number of partitions in x and y range.
PARTITION-NUM-X 2
PARTITION-NUM-Y 2
#
```

We would now have a simulation are of size (100 * 100). We would also have 4 partitions (each partition is represented by a single entity) in the simulation. So one partition would encompass the square area represented by the coordinates (0, 0), (49, 0), (0, 49), and (49, 49).

1.2 Layered Structure

Since we are building GloMoSim using a layered approach, we would like to have the ability to rapidly integrate models developed at different layers by different people. Hence the simple approach here would seem to be that each layer in the simulation would be represented by a different Parsec entity. We would still have the same problem that we had previously. As the number of layers in the simulation increases, the number of entities in the simulation would also increase. This would lead to scalability and performance problems in the simulation. But this is not as dramatic since there are only a few layers in the simulation. But there are other reasons why we need to aggregate the layers into a single entity.
Often times, different layers of the simulation need to access certain common variables. For example, the upper layers of the simulation need to use the CPU when they are executing any instructions. But CPU is a shared resource among these layers. Hence, before executing any instructions a layer has to make sure that the CPU is free. Hence the upper layers need to have access to common variables which will provide information about the state of the CPU. If these layers are kept as different entities in the simulation we don't have a way of accessing shared variables. Besides we don't want to use any global variables as they can create problems for parallel execution.

If the layers are kept as different entities, each layer also has to explicitly keep track of the "ename" value for the upper and lower layers. These "ename" values are needed for message passing among the various layers. For parallel conservative runtime, each entity also has to specify the source and destination set as well as the lookahead values for the entity. Specifying lookahead for an entity can become very complicated. All this creates additional work for the developer who is basically interested in modeling a particular network protocol.

For these reasons, we decided to integrate the various GloMoSim layers into a single entity. Each entity now encompasses all the layers of a simulation. Instead each layer is now implemented as functions in

the new design. We provide an initialization function that will be called for each layer of each node at the beginning of the simulation. We provide functions that can be used to send messages between the layers. When a layer receives a particular message, it will automatically invoke a function that is provided by the developer of that particular layer. And based on the contents of the message, the function can then execute the appropriate instructions. At the end of the simulation, a function is also called for each layer of each node. A layer can use this function to collect any relevant statistics. An actual implementation of a particular layer will be shown shortly.

2. Directory Structure of GloMoSim.

After downloading and upzipping GloMoSim, it should contain the following directories:

/applicaiton contains code for the application layer
/bin for executable and input/output files
/doc contains the documentation
/include contains common include files
/mac contains the code for the mac layer
/main contains the basic framework design
/network contains the code for the network layer /radio contains the code for the physical layer /transport contains the code for the transport layer

You have to compile the files from the main/ directory.

- Run "make depend" to create list of depndencies in the Makefile.
- Make sure that the right path for the Parsec compiler is specified in the Makefile for the "PAR" variable. o Run "make" to create the executable

You can also use Makent.bat for compiling in batch mode (for NT).
To run the simulation you have to go to the bin/ directory. The executable is called "Sim". It taked only one command line paramter, which is an input file. An example input file is CONFIG.IN in bin/ directory. Run "Sim CONFIG.IN" to run the program. A file called "GLOMO.STAT" is produced at the end of the simulation and contains all the statistics generated by the simulation. Make modifications to CONFIG.IN to vary the parameters for running the simulation.

3. Description of input file.

This section explains the various parameters which are part of the input file supplied to the GloMoSim executable. In the input file anything following a "#" is treated as a comment. Note that some parameters presented in this section may not be valid in the latest GloMoSim library as we keep updating the library to be configured easily. The "config.in" file included in the distribution should be the most up-to-date configuration file and used as a template file.

The following two parameters stand for the physical terrain in which the nodes are being simulated. For example, the following represents an area of size 100 meters by 100 meters. All range parameters are in terms of meters.

```
#
# Terrain Area we are simulating.
TERRAIN-RANGE-X 100
TERRAIN-RANGE-Y 100
#
```

The following parameter represents the power range of wireless nodes in the simulation. For example, a node can reach any other node within 50 meters of its position.

```
#
POWER-RANGE 50
#
```

The following is a random number seed used to initialize part of the seed of various randomly generated numbers in the simulation. This can be used to vary the seed of the simulation to see the consistency of the results of the simulation.

```
#
SEED 1
#
```

The following parameter represents the maximum simulation time. The numbered portion can be followed by optional letters to modify the simulation time.

For example:
100NS - 100 nano-seconds
100MS - 100 milli-seconds
100S - 100 seconds
100 - 100 seconds (default case)
100M - 100 minutes
100H - 100 hours
100D - 100 days

```
#
SIMULATION-TIME 100M
#
```

The following parameter represents the number of nodes being simulated.

```
#
NUMBER-OF-NODES 12
```

#

Conclusion:

Here the glomo simulator has been evaluated with the range of small time manipulation.

So every part of delivery standards can be modified easily,.