آموزش برنامه نویسی تلفن های همراه
به زبان J2ME در IDE NetBeans

نویسندگان :

مهندس ایمان اشکاوند راد

مهندس حسن فرزانه

سال ۱۳۸۶

امام علی (ع) فرمودند :

زکات علم نشر و گسترش آن می باشد

این کتاب الکترونیکی کاملاً رایگان می باشد

و

به تمامی ایرانیان عزیز و فارسی زبانان تقدیم می گردد

Http://www.Ashkavand.ir

چکیده

آموزش برنامه نویسی تلفن های همراه
به زبان J2ME در IDE NetBeans

این PDF شامل دو فصل از مباحث برنامه نویسی تلفن همراه به زبان J2ME است . بعد از مطالعه این کتاب شما با انواع زبانهای برنامه نویسی در سیستم عامل سیمبین و به ویژه زبان J2ME آشنا میشوید. مطالعه این کتاب همچنین شما را با محیط برنامه نویسی نت بینز ۵ و انواع کامپوننت ها و نحوه برنامه نویسی در آن و قابلیت های آن برای برنامه نویسی به زبان J2ME آشنا خواهد کرد. در ادامه نیز با نحوه برنامه نویسی پایگاه داده RMS در J2ME آشنا خواهید شد.

واژه‌های کلیدی

سیستم عامل سیمبین، J2ME، نت بینزRMS ، NetBeans , Symbian , جاوا
Mobile Programming

فصل اول - آشنایی با محیط برنامه نویسی J2ME و IDE Netbeans

فصل اول - آشنایی با محیط برنامه نویسی J2ME و IDE Netbeans

۱-۱- آشنایی با محیط های برنامه نویسی جاوا

امروزه، با وجود ایستگاه‌های کاری، که دارای پردازنده‌های سه گیگا هرتزی و RAM های یک گیگابایتی هستند، محیط‌های قدرتمند می‌توانند بدون وارد ساختن بار اضافی بر سخت‌افزار اجرا گردند. در نتیجه این پیشرفت، طی چند سال گذشته شاهد ظهور محصولات قابل توجهی، از قبیل ویژوال استودیو دات نت[1]، ایکلیپس ۳,۰[2]، نت بینز ۵,۰[3] و ... بوده‌ایم که سطح IDE ها را بالا برده‌اند.

چهار شرکت پیشرو برای توسعه جاوا و ابزارهای همراه آن عبارتند از :

- شرکت سان میکرو سیستم
- شرکت آی بی ام
- شرکت اراکل
- شرکت بورلند

[1] Visual Studio.Net
[2] Eclipse 3.0
[3] Netbeans 5.0

شکل ۱-۱- محیط های برنامه نویسی جاوا

این محصولات به شکلی جالب توجه، سمبل‌های IDE متفاوت را به کار می‌برند. محصول ایکلیپس ۳,۰ آی بی ام، مبتنی بر محصول رایگان کدباز است که مقبولیت زیادی در بین توسعه‌گران جاوا دارد. محصول سان مبتنی بر پلتفرم IDE کد باز رقیب، یعنی نت بینز است و محصولات اراکل و بورلند بر مبنای رابط‌های کاربر اختصاصی ساخته شده‌اند.

من برای انجام این پروژه محیط برنامه نویس نت بینز ۵ را انتخاب نمودم که توسط شرکت سان میکرو سیستم ارائه شده است و دارای قابلیت های خوبی برای برنامه نویسی به زبان J2ME می باشد.

علاوه بر یک محیط برنامه نویسی برای اجرای برنامه ها به زبان J2ME نیاز به نصب یک SDK نیز داریم. SDK حاوی API ها، کتابخانه‌ها، شبیه‌ساز و دیگر ابزار کار با سیستم عامل مربوطه هستند. برای سیمبین سری ۶۰ می‌توانیم از SDK های ویرایش 1.2 یا 2.x استفاده نماییم.

ملزومات لازم برای ایجاد محیط برنامه نویسی J2ME

در زیر ملزومات لازم برای ایجاد یک محیط برنامه نویسی به زبان J2ME نام برده شده است.

۱. Java Standard Development Kit (JDK™) version 5.0

۲. (WTK2.0) Emulator Wireless Toolkit 2.0

۳. IDE NetBeans 5.0

۴. NetBeans Mobility 5.0 Pack

برای دانلود این ملزومات می توانید به سایت www.netbeans.org مراجعه نمایید. پس از تهیه این نرم افزارها به ترتیب ذکر شده اقدام به نصب آن ها نمایید.

۱-۲- طراحی ویژوال برنامه های MIDP

یک VMD [1] شما را قادر می سازد که به صورت ویژوال به طراحی روند جریان برنامه هایتان و همچنین رابط کاربر گرافیکی (GUI) بپردازید. به عنوان مثال شما می توانید با کشیدن و رها کردن اشیا بر روی پنجره های Flow Designer و Screen Designer یک محیط را طراحی نمایید و محیط برنامه نویسی برای شما به طور خودکار کدهای جاوای آن اشیا را تولید خواهد نمود.

برای استفاده از حالت VMD، شما باید با فایل هایی کار نمایید که به وسیله الگوهای VMD تولید می شوند و در قسمت ایجاد فایل جدید [2] به صورت ویزارد وجود دارند. شما نمی توانید از حالت VMD برای اصلاح فرم های GUI یی که خارج از IDE تولید شده اند، استفاده نمایید. همچنین اگر شما کد تولید شده در محیط VMD را در کد منبع تغییر دهید، شما دیگر نمی توانید در حالت ویژوال به کار خود ادامه دهید.

VMD ابزارهای زیر را برای ساده تر شدن فرآیند طراحی برنامه های MIDP شما فراهم نموده است که این ابزارها عبارتند از:

- **Flow Designer:** این محیط فضای کاری اصلی VMD می باشد. در این فضا شما شاهد نمایی کلی از کامپوننت های صفحه [3] و دیگر کامپوننت های UI و همچنین نحوه اتصال فرمان [4] ها به کامپوننت ها هستید.
- **Screen Designer:** این محیط شما را قادر به طراحی و اصلاح صفحه های داخل برنامه می نماید.

---

[1] Visual Mobile Designer
[2] New File
[3] Screen
[4] Command

- **پنجره Inspector:** این پنجره یک ساختار درختی از همه کامپوننت های موجود در آرشیو برنامه جاری را نمایش می دهد که شامل صفحه ها، فرمان ها و منابع[1] می باشد.
- **پالت Component:** این پالت شامل همه اشیاء جاوا است که شما می توانید به برنامه تان اضافه نمایید.
- **مدیریت پالت Component:** شما را قادر می سازد که صفحه ها و اشیاء خودتان را به پالت کامپوننت اضافه نمایید.
- **Projects Sheet:** این صفحه به نمایش تنظیمات قابل اصلاح، برای کامپوننت انتخاب شده جاری می پردازد.

۱-۳- ایجاد یک پروژه جدید در حالت VMD

برای این منظور باید مراحل زیر را انجام دهید:

۱. از منوی فایل گزینه New Project (Ctrl-Shift-N) را انتخاب نمایید. در پنجره باز شده در قسمت Categories گزینه Mobile ودر قسمت Projects گزینه Mobile Application را انتخاب نموده وسپس دکمه Next را کلیک نمایید.

۲. سپس یک نام مانند myHello را در فیلد Project Name وارد نمایید. سپس شما می توانید مسیر ذخیره شدن پروژه را تغییر داده و در هر مسیر دلخواه از سیستم خود آن را قرار دهید.

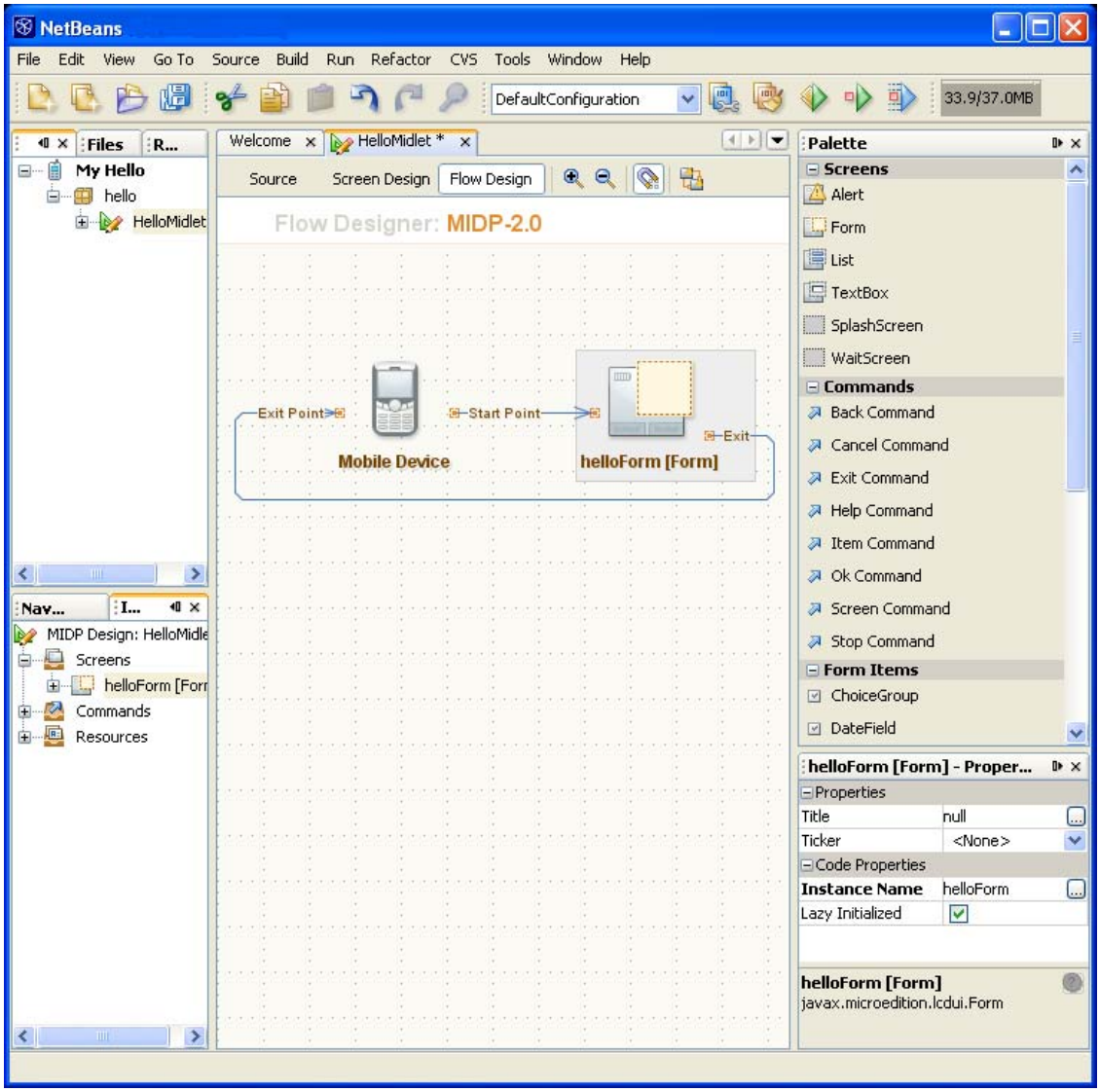
۳. چک نمایید که دو چک باکس Set as Main Project و Create Hello MIDlet تیک خورده باشند (البته این دو گزینه به طور پیش فرض انتخاب شده اند).

۴. سپس J2ME Wireless Toolkit 2.2 را به عنوان Emulator Platform انتخاب نموده و نسخه MIDP و CLDC را با توجه به نوع وسیله خود تعیین نمایید.

۵. کلید Finish را کلیک نمایید. محیط برنامه نویسی برای شما در مسیر انتخاب کرده در مراحل قبل یک پوشه با نام myHello/... ایجاد می نماید. این پوشه پروژه شامل همه فایل های منبع و متادیتا[2]های پروژه شما از قبیل Project Ant Script می باشد.

برنامه خودش را در پنجره Flow Design از حالت VMD به نمایش می گذارد.

[1] Resource
[2] MetaData



شکل ۱-۲- اجرای یک پروژه جدید در حالت VMD

اصلاح کد منبع جاوا

شما همچنین می توانید متن نمایش داده شده به وسیله MIDlet را اصلاح نمایید. برای این منظور این اعمال را انجام دهید:

۱. بر روی Screen Design از Toolbar کلیک نمایید. با این عمل پنجره Screen Designer باز می شود و برای شما محتویات فرم helloForm را نمایش می دهد.

۲. شما با دابل کلیک بر روی متن "Hello World!" در پروژه ایجاد کرده آن را به حالت انتخاب در آورده و هر متن دلخواهی را می توانید در داخل آن وارد نمایید.

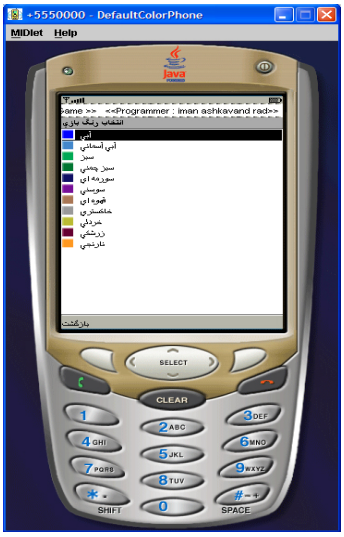
۱-۴- اجرای یک برنامه MIDP

هنگامی که شما یک برنامه MIDP را اجرا می نمایید. محیط برنامه نویسی، عملیات کامپایل، پیش شناسایی و ساختن برنامه را بر طبق پیکربندی های نصب شده برای پروژه جاری به عنوان یک MIDlet اجرا می نماید. خروجی در یک ایمولاتور[1] نمایش داده می شود.

همچنین شما می توانید پروژه اصلی[2] یا هر پروژه مجزای دیگری که شامل یک کلاس قابل اجرا باشد را اجرا نمایید.

برای اجرای یک پروژه در نت بینز شما می توانید یکی از چهار عمل زیر را انجام دهید:

- از طریق منوی Run و انتخاب گزینه Run Main Project
- فشردن کلید F6
- از طریق فشردن آیکون اجرای پروژه اصلی( )
- از طریق راست کلیک بر روی پروژه در پنجره Project و انتخاب گزینه Run Project



شکل۱-۳ - اجرای یک پروژه در ایمولاتور

۱-۵- اجرای پروژه با یک ایمولاتور متفاوت

برای این منظور شما باید ابتدا بر روی پروژه راست کلیک نموده و گزینه Quick Run With را انتخاب نمایید. سپس از لیست کشویی که شامل ایمولاتور های نصب شده می باشد یکی را انتخاب نمایید.

پروژه برای اجرا از طریق توزیع فایل JAR استفاده می نماید و نیازی به ساخت دوباره پروژه قبل از اجرای آن نیست.

[1] Emulator
[2] Main Project

۱-۶- اضافه کردن یک MIDlet

برای اضافه کردن یک MIDlet جدید به پروژه باید سه عمل زیر را انجام دهید:

۱. ابتدا یک پروژه را انتخاب نموده و از منوی فایل گزینه New File را انتخاب نمایید.
۲. در پنجره باز شده از قسمت Categories گزینه MIDP و در قسمت FileType گزینه MIDlet را انتخاب نمایید.
۳. سپس فایل الگوی خود را بر حسب نیازهایتان به آن اختصاص دهید.

۱-۷- MIDP Screens

با توجه به صفحه های نمایش کوچک و حافظه محدود و قدرت پردازش بر روی وسایل هندهلد[1]، MIDP از یک کلاس خاص که Displayable نامیده می شود برای نمایش یک UI روی وسایل به جای سیستم پنجره بندی معمولی استفاده می کند.

یک Displayable از نظر رفتاری مشابه با یک پنجره است، با این تفاوت که فقط یک Dispalyable می تواند در هر زمان قابل نمایش باشد. در یک برنامه MIDP، کامپوننت های Displayable گوناگون مانند یک دسته کارت برروی یکدیگر قرار می گیرند.کاربر مجموعه کامپوننت های Displayable را یکی یکی پشت سر می گذارد تا عملیات کامل شود. جابه جایی بین دسته Displayable ها می تواند خطی یا غیر خطی باشد، اما باید برای کاربر روشن باشد که برای جابه جایی به Displayable بعدی چه کاری باید انجام دهد.

کلاس Displayable دو زیر کلاس دارد: Canvas و Screen. شش نوع از کامپوننت هایی که از Screen مشتق شده اند عبارتند از:

- Alert
- Form
- List
- TextBox
- SplashScreen
- WaitScreen

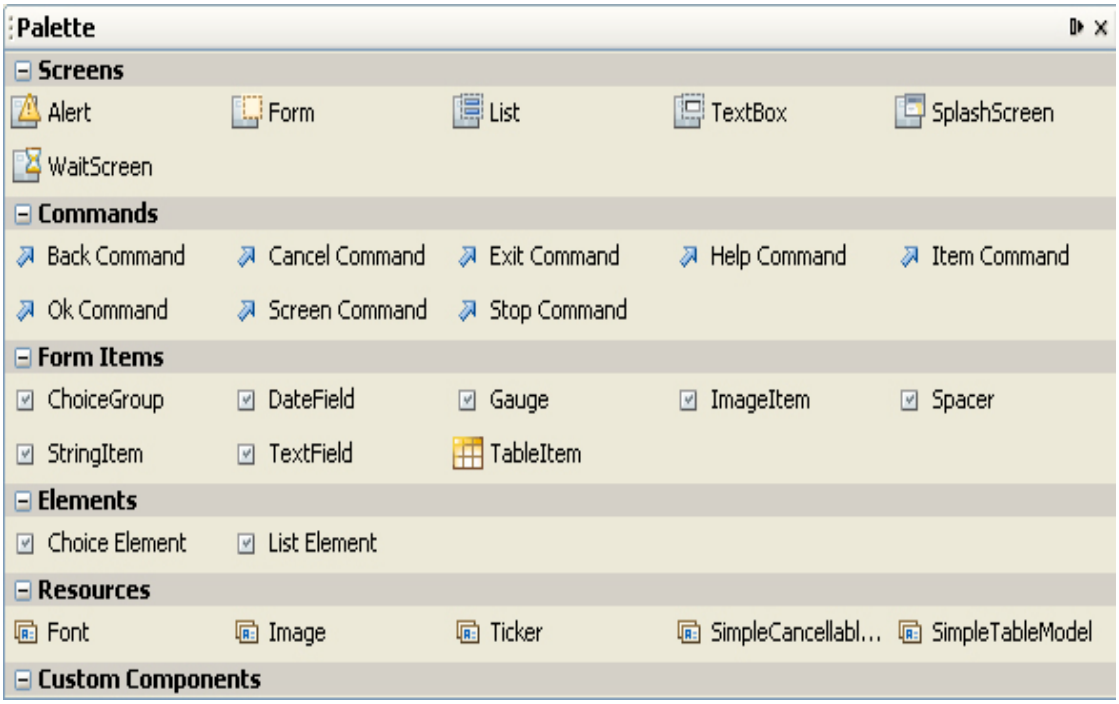
شما با Canvas و همه کلاس های Screen بر حسب نوع رفتار شان با بیرون، رفتار می کنید. طراحی UI برای صفحه های Displayable به سادگی امکان پذیر است. فقط باید هر صفحه را به یک وظیفه خاص محدود نمایید و برای جابه جایی بین صفحات از فرمان ها استفاده نمایید.

[1] Device Hand Held

وقتی شما از حالت VMD برای پروژه خود استفاده می نمایید، کار با کامپوننت های صفحه[1] بسیار راحت است و فقط شما کافی است که کامپوننت مورد نظر خود را بر روی صفحه Flow Designe قرار دهید و تنظیمات آن را در پنجره properties انجام دهید. کد مربوط به آن به طور خودکار تولید خواهد شد.

Canvas برروی پالت قرار ندارد زیرا برای استفاده از یک Canvas، شما باید با پیاده سازی متد کلاس paint() در زیر کلاسش، آن را فراهم نمایید. زیرا طراح UI نمی تواند این کار را انجام دهد.

در شکل ۱-۴ پالت MIDP را به همراه کامپوننت های آن مشاهده می نمایید.



شکل۱-۴– پالت کامپوننت های MIDP

۱-۸- معرفی کامپوننت های قسمت Screens

۱-۸-۱-کامپوننت Alert

یک Alert عبارت است از یک متن به همراه یک آیکون که به کاربر نشان داده می شود و برای مدت زمان مشخصی قبل از بسته شدن یا تا زمانی که کاربر صفحه را ببندد، منتظر می ماند. Alert اصولاً برای گزارش خطاها و سایر شرایط استثنایی دیگر به کار می رود. شما کنترل محدودی روی لایه های این کامپوننت

[1] Screen

دارید. یک Alert می تواند از یک AlertType برای مشخص کردن نوع هشدار و پخش یک صدای مناسب در زمان باز شدن آن استفاده نماید.

تعدادی از متدهای مهم این کامپوننت عبارتند از:

- getDefaultTimeout: زمان پیش فرض برای نمایش یک Alert را بر حسب میلی ثانیه بر می گرداند.
- getTimeout: مقدار زمانی را که این Alert نمایش داده خواهد شد، را بر می گرداند.
- getString: رشته متنی را که برای نمایش در Alert استفاده شده را بر می گرداند.
- setTimeout: مقدار زمان مورد نظر برای نمایش Alert را مقدار دهی می نماید.
- setImage: عکس مورد استفاده در Alert را مقدار دهی می نماید.
- setString: متن مورد استفاده در Alert را مقدار دهی می نماید.

تکه کد زیر نحوه ایجاد وکاربرد یک Alert را نمایش می دهد.

```
// Popping an Alert...
// Parameters..
// 1. Title of the Alert..
// 2. Text of Alert.
// 3. Image if required.
// 4 Type of Alert (INFO, WARNING,ERROR, CONFIRMATION, ALARM).


Alert alert = new Alert ("Warning", "You have entered Wrong serial number",
                        null, AlertType.WARNING);
// Making this alert a Modal alert..
    alert.setTimeout(Alert.FOREVER);
//  or else...
// Making this alert disappear after certain milliseconds..
    alert.setTimeout(10);
```

۱-۸-۲-کامپوننت Form

فرم تنها کامپوننت MIDP است که می تواند شامل کامپوننت های دیگر شود. فرم یک کامپوننتی است که بیشتر در ایجاد یک UI استفاده می شود.

شما به طور تئوری می توانید کامپوننت های زیادی در یک فرم آن گونه که می خواهید قرار دهید، اما با توجه به ناحیه نمایش کوچک تلفن های همراه شما باید هر فرم را به یک وظیفه خاص محدود نمایید. همچنین طول یک صفحه را نیز باید محدود نمایید که کاربر نیازی به پیمایش بیشتر از سه یا چهار صفحه بر روی وسیله مقصد نداشته باشد. این مورد کارایی وسیله را افزایش می دهد و آن را برای پیمایش کاربر راحتتر می سازد.

در برنامه نویسی MIDP 1.0 توسعه دهنده برنامه، کنترلی بر روی آرایش[1] کامپوننت در یک فرم ندارد. تنها چیزی که می توانید درباره مکان کنترل کنید، نظم کامپوننت های گوناگون در فرم است. تقریباً مشابه آرایش عمودی در J2SE است.

در MIDP 2.0 شما تا حدی بر روی آرایش کامپوننت ها بر روی فرم کنترل دارید. نت بینز یک خاصیت به نام Layout دارد که به شما اجازه می دهد تا ترازبندی افقی، عمودی و نحوه پر کردن[2] کامپوننت را مشخص نمایید.

تعدادی از متدهای مهم این کامپوننت عبارتند از :

- addCommand: یک فرمان را به فرم اضافه می نماید.
- Append : یک عکس یا یک رشته متنی یا یک عنصر را به فرم اضافه می نماید.
- Delete(int itemnum): عنصری راکه شماره آن به عنوان ورودی داده شده است را پاک می نماید.
- deleteAll : همه عناصر روی فرم را پاک می نماید.

کد زیرنحوه ایجاد کامپوننت Form و اضافه کردن آیتم ها را به آن را نشان می دهد.

```
// Declaring the form...
Form myform = null;
...
// Declaring the Display and initializing it as null...
private Display show = null;
...
// Getting the Display unique to this MIDlet...
show = Display.getDisplay(this);
// Initializing the form with a string title...
myform = new Form("User Interface - TextField");
// Declaring and initializing a TextField...
tx = new TextField("MyField","Type here...",70,0);
// Adding the TextField to the form...
myform.append(tx);
// Showing the form, which is a Displayable object, on the screen...
show.setCurrent(myform);
```

[1] Layout
[2] Fill

۱-۸-۳-کامپوننت List

از این کامپوننت برای نمایش و انتخاب یک لیست در سه حالت زیر استفاده می شود:

- IMPLICT: از این حالت می توان برای ایجاد یک منو استفاده نمود.
- EXCLUSIVE: از این حالت می توان برای ایجاد یک لیست انتخاب منفرد استفاده نمود.
- MULTIPLE: از این حالت برای ایجاد حالت بیش ازیک انتخاب استفاده می گردد.

تعدادی از متدهای این کامپوننت عبارتند از:

- getImage(int elementnum) : عکس قسمتی از لیست را که به وسیله پارامتر elementnum مشخص شده را بر می گرداند.
- getString(int elementnum) : متن قسمتی از لیست را که به وسیله پارامتر elementnum مشخص شده است را بر می گرداند.
- getSelectedIndex(): شماره آیتمی از لیست را که انتخاب شده است بر می گرداند.
- Append(String stringpart,Image imagepart) : این گزینه یک آیتم به List اضافه می کند.
- isSelected: یک مقدار بولین را بر می گرداند که نشان می دهد آیا آیتم مورد نظر انتخاب شده است.

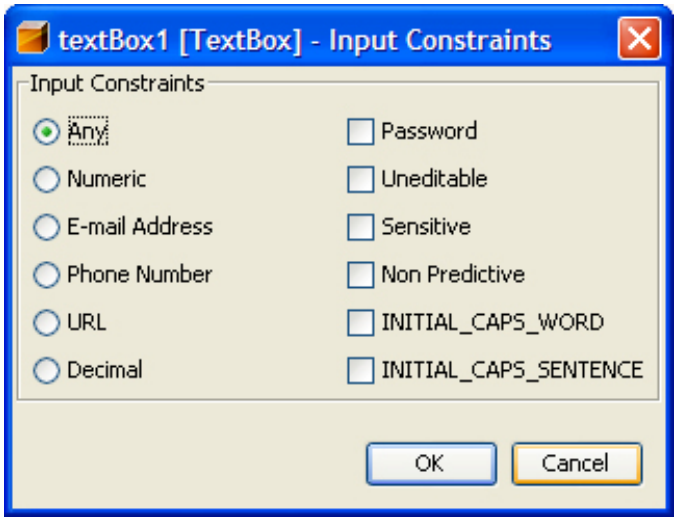
کد زیر نحوه ایجاد و استفاده از کامپوننت List را نشان می دهد.

```
// Declaring a variable of the type List..
List menu = null;
// Initializing the variable ...
// Parameter 1 --- Label of the List.
// Parameter 2 --- List Type IMPLICIT. (Selection by scrolling)

  menu = new List("Various Options..",List.IMPLICIT);

// Adding the List Items..
// Parameter 1 --- Label.
// Parameter 2 --- Image if required.

  menu.append("TextField",null);
  menu.append("Ticker",null);
  menu.append("Alert",null);

// Adding action Listener to the List..

  menu.setCommandListener(this);

// Making it the current Display.

  display.setCurrent(menu);

// Handling the events on the List.
 public void commandAction(Command c, Displayable d)
{
    List down = (List)display.getCurrent();
    switch(down.getSelectedIndex()) {
    case 0: System.out.println ("Text Field...");
            break;
    case 1: System.out.println ("Ticker...");
            break;
    case 2: System.out.println("Alert...");
            break;
}
```

۱-۸-۴-کامپوننت TextBox

TextBox یک کامپوننتی است که کاربر می تواند متن مورد نظر خود را در آن وارد نموده و ویرایش نماید. شما می توانید حداکثر اندازه آن را مشخص نمایید و یک مقدار پیش فرض برای خاصیت String آن مشخص نمایید. اگر متنی که در Textbox قرار دارد بیشتر از این باشد که در یک صفحه نمایش داده شود، کاربر می تواند برای دیدن یا ویرایش هر قسمت متن آن را پیمایش نماید.

TextBox قواعدی را استفاده می کند که به برنامه کاربردی اجازه محدود کردن ورودی کاربر، در مواردی مانند آدرس پست الکترونیکی، مقادیر عددی، شماره تلفن، URL و کلمه عبور را می دهد.



شکل ۱-۵- محدود کردن ورودی کاربر در TextBox

تکه کد زیر نحوه ایجاد یک Textbox را نشان می دهد.

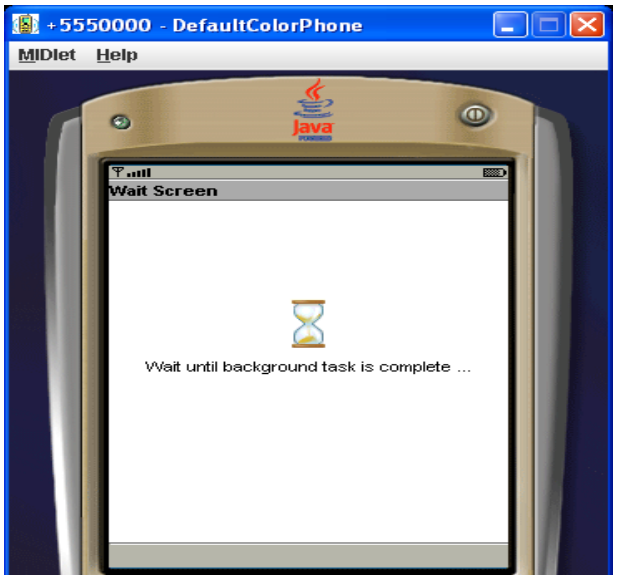
```
String s = "Hello From J2ME";
...
private Display ourDisplay;  // Declaring the display
private Form ourForm = null;
...
ourDisplay = Display.getDisplay(this);
...
ourForm = new Form("Our First");
TextBox ourBox = new TextBox("J2ME Application", s, 256, 0);
ourForm.append(ourBox);
...
ourForm.setListener(this);

ourDisplay.setCurrent(ourForm);
```

۱-۸-۵-کامپوننت SplashScreen

این کامپوننت شبیه به کامپوننت Alert می باشد، که به وسیله آن می توانید یک عکس با فرمت PNG و یک متن کوتاه را برای مدت زمان مشخص شده که توسط خاصیت Timeout در SplashScreen تعیین می شود، نمایش داد. مقدار پیش فرض این خاصیت ۵۰۰۰ میکروثانیه است. همچنین بعد از زمان تعیین شده برنامه به طور خودکار به اجرای عملی که درخاصیت Default Action تعیین کردیم می پردازد. مثلاً روانه شدن به یک فرم دیگر.

۱-۸-۶-کامپوننت WaitScreen

از این کامپوننت برای مواقعی که بخواهیم عمل زمان بری را انجام دهیم و کاربر را از این عمل مطلع کنیم، استفاده می شود. مثلاً نمایش متن " ... Wait until background task is complete" به همراه یک تصویر ساعت شنی.



شکل ۱-۶- استفاده از کامپوننت WaitScreen

۱-۹- معرفی فرمان های موجود در پالت

در این قسمت ما دارای هشت نوع فرمان مختلف برای کاربردهای خاص هر کدام هستیم. این انواع شامل Back ، Cancel ، Exit، Help، Ok، Item، Screen و Stop می باشد. شما می توانید با توجه به نیاز برنامه خود هر کدام از این انواع را که خواستید بر روی فرم خود کشیده و با کلیک بر روی Edit برای آن ارتباط لازمه را مشخص نمایید یا با مراجعه به قسمت سورس برنامه در تابع Command Action برای آن فرمان کدهای لازم را بنویسید.

```
//...
private Command exitCommand = new Command("Exit", Command.EXIT, 1);
private Command backCommand = new Command("Back", Command.BACK, 1);
//...
public void commandAction(Command command, Displayable displayable){
  if (displayable == helloForm) {
        if (command == exitCommand) {
                exitMIDlet();

        } else if (command == backCommand)

           getDisplay().setCurrent(get_helloForm());
     }
//...
```

سه پارامتری که برای به وجود آوردن یک شی از نوع فرمان مورد نیاز است عبارتند از:

- Lable: برچسب فرمان را تعیین می کند.
- Position: مکان فرمان را مشخص می کند.
- Priority: اولویت فرمان را مشخص می کند.

تعدادی از متدهای مهم این کامپوننت عبارتند از:

- getCommandType: نوع فرمان را بر می گرداند. مثلاً از نوع Command.ok است یا از نوع Command.Back و غیره.
- getlable: برچسب فرمان را برمی گرداند.
- getpriority: اولویت فرمان را برمی گرداند.

۱-۱۰- معرفی کامپوننت های قسمت Form Items

۱-۱۰-۱-کامپوننت ChoiceGroup

با استفاده از این کامپوننت کاربر می تواند یک یا چندگزینه رشته ای را انتخاب نماید. هر رشته می تواند یک تصویر اختیاری نیز داشته باشد. دکمه های رادیویی معمولاً برای انتخاب های منفرد و چک باکس ها برای چندین انتخاب همزمان استفاده می گردند. با توجه به پارامتر ChoiceType آن، اگر Exclusive باشد یک چک باکس و اگر Multiple باشد یک دکمه رادیویی داریم.

تعدادی از متدهای این کامپوننت عبارتند از:

- Append(String stringpart , Image imagepart) : یک آیتم به Choicegroup اضافه می نماید.
- Delete(int elementnum) : آیتمی را که elementnum به آن اشاره می کند را پاک می نماید.
- getImage(int elementnum) : عکس قسمتی از Choicegroup را که elementnum به آن اشاره می کند را بر می گرداند.
- getSelectedIndex : شماره آیتمی از Choicegroup را که انتخاب شده است را بر می گرداند.

تکه کد زیر چگونگی ایجاد یک دکمه رادیویی را توسط این کامپوننت نشان می دهد.

```
// Declaring Form and ChoiceGroup objects and initializing them to null...
Form ui_holder = null;

ChoiceGroup radiobutton_type = null;

String[] name = {"a","b","c"};
Image[] img = null;

// Initialize the ChoiceGroup..
// Parameter 1 ---   Title
// Parameter 2 ---   Type of ChoiceGroup (Exclusive for RadioButtons)
// Parameter 3 ---   Label of the radio buttons displayed
// Parameter 4 ---   Images for the radio button label if required

radiobutton_type = new ChoiceGroup("Choices..",ChoiceGroup.EXCLUSIVE,name,img);

// Adding the ChoiceGroup to the Form...
ui_holder.append(radiobutton_type);

// Event Handling..
// To get the String of the selected radio button...
radiobutton_type.getString(radiobutton_type.getSelectedIndex());
```

۱-۱۰-۲-کامپوننت DateField

با استفاده از این کامپوننت شما می توانید فیلد تاریخ و زمان قابل ویرایش در اختیار داشته باشید. شما می توانید فقط تاریخ یا زمان و یا هر دو را نمایش دهید.

۱-۱۰-۳-کامپوننت Gauge

از این کامپوننت برای نمایش یک محدوده ازمقادیر که کاربر می تواند مشخص نماید استفاده می شود. همچنین شما می توانید با استفاده از خاصیت Maximum Value مقدار حداکثر این کامپوننت و توسط خاصیت value مقدار جاری این کامپوننت را تعیین نمایید.

تعدادی از متدهای مهم این کامپوننت عبارتند از:

- setValue(int value): این گزینه Gauge را با مقدار value مقدار دهی می کند.
- setMaxValue(int value): ماکزیمم مقدار Gauge را مشخص می کند.
- getValue(): مقدار Gauge را بر می گرداند.
- getMaxValue(): ماکزیمم مقدار Gauge را بر می گرداند.

تکه کد زیر چگونگی ایجاد یک کامپوننت Gauge از نوع محاوره ای را نشان می دهد.

```
private Display display = null;
Form ui_holder = null;
...
// Declaring variable for the Gauge class...
Gauge gaugeUI;
...
display = Display.getDisplay(this);
...
ui_holder = new Form("User Interface - Gauge ");
gaugeUI = new Gauge("Values..",true,10,0);
ui_holder.append(gaugeUI);
...
display.setCurrent(ui_holder);
...

// Event handling for the OK button..
if ( c == ok ) {
int i = gaugeUI.getValue();

display.setCurrent(ui_holder);
}
...
```

۱-۱۰-۴-کامپوننت ImageItem

این کامپوننت یک محل برای قرارگیری عکس با فرمت PNG است که می تواند شامل یک برچسب باشد و می توان تا اندازه ای به وسیله خاصیت Layout، محل عکس را روی صفحه کنترل نمود.

۱-۱۰-۵-کامپوننت Spacer

از این کامپوننت برای جدا نمودن وایجاد فضای خالی (یک خط) بین کامپوننت ها استفاده می شود.

۱-۱۰-۶-کامپوننت StringItem

یک فیلد متنی غیر قابل ویرایش که می تواند یک برچسب داشته باشد. برچسب و محتوای آن می تواند به وسیله برنامه کاربردی ویرایش گردند. تکه کد زیر چگونگی ایجاد یک StringItem را نشان می دهد.

```
// Declaring variable for StringItem class.
StringItem string_item = null;
// Initializing the variable

// parameter 1 .. Label to be displayed
// parameter 1 .. Text along with the Label.

string_item = new StringItem("User Entered ..", "");


// Adding String item to the form (place holder)
ui holder.append(string item);


// Changing the Textual context of the StringItem

  string_item.setText(textcheck.getString());
```

۱-۱۰-۷-کامپوننت TextField

این کامپوننت یک فیلد متنی قابل ویرایش است که می تواند یک برچسب و یک متن اولیه داشته باشد. حداکثر تعداد کارکترها را می توانید مشخص نمایید. TextField قواعدی برای فیلدهای خاص مانند آدرس پست الکترونیکی، مقادیر عددی، شماره تلفن، URL و کلمه عبور دارد. کامپوننت های TextField گوناگون می توانند برای ورودی کاربر به فرم اضافه شوند.

بعضی از محدودیت های قابل اعمال توسط این کامپوننت عبارتند از:

- TextField.ANY : کاربر اجازه دارد هر متنی دلخواهی را وارد نماید.
- TextField.EmailAddr : کاربر اجازه دارد فقط آدرس ایمیل را وارد کند.
- TextField.Password : متن وارد شده قابل مشاهده نیست.
- TextField.PhoneNumber : برای وارد کردن شماره تلفن قالب بندی شده است.

تعدادی از متدهای این کامپوننت عبارتند از:

- Getstring : محتویات فیلد متنی را که یک رشته است برمی گرداند.
- Getconstraint : محدودیت اعمال شده روی فیلد متنی را بر می گرداند.
- Setstring(String text) : محتویات متن داده شده به آن را به جای محتویات فیلد متنی جایگذاری می کند.
- getMaxsize() : ماکزیمم تعداد کاراکترهایی که در فیلد متنی می توانند قرار بگیرند را بر می گرداند.

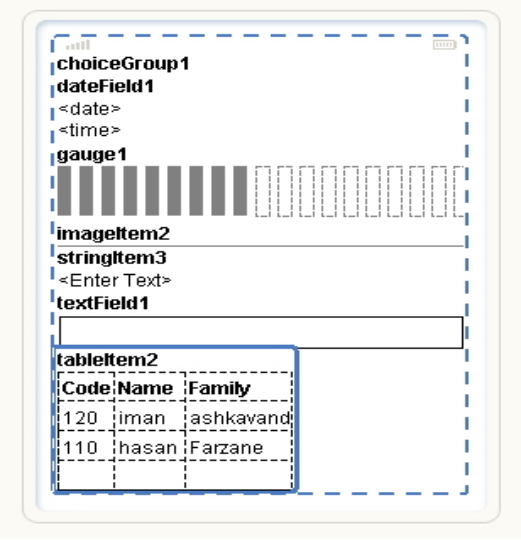
تکه کد زیر چگونگی ایجاد یک TextField را نشان می دهد.

```
// Declaring a variable of the type TextField
 TextField textcheck = null;

 textcheck = new TextField("Enter the Text Here..","",50,0);

// Adding TextField to the form (place holder)
  ui_holder.append(textcheck);

// Event Handling..

public void commandAction(Command c, Displayable d) {
// Event handling for the Button
  if ( c == ok )
  {
   System.out.println(textcheck.getString());
  }
 }
```

۱-۱۰-۸-کامپوننت TabelItems

از این کامپوننت برای نمایش اطلاعات در داخل جدول می توان استفاده نمود. با دابل کلیک بر روی این کامپوننت صفحه ای برای ورود و اصلاح اطلاعات به درون جدول باز می شود. همچنین شما می توانید تعداد سطر یا ستون جدول را نیز تعیین نمایید.

در شکل ۱-۷ تمامی کامپوننت های Form Item را بر روی یک فرم مشاهده می نمایید.



شکل ۱-۷- کامپوننت های قسمت Form Items بر روی فرم

۱-۱۱-کامپوننت های بخش Elements

۱-۱۱-۱-کامپوننت Choice Element

از این کامپوننت برای قرار گرفتن در یک ChoiceGroup استفاده می گردد و می توان چندین کامپوننت از این نوع را به میزان لازم برروی ChoiceGroup مورد نظر قرار داد.

۱-۱۱-۲-کامپوننت List Element

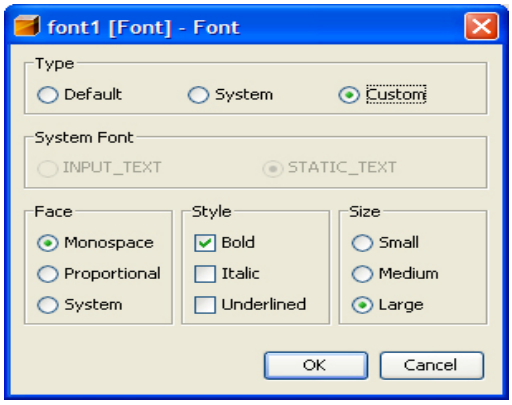
از این کامپوننت برای قرار گرفتن در یک List استفاده می گردد و می توان چندین کامپوننت از این نوع را به میزان لازم برروی لیست مورد نظر قرار داد.

شکل۱-۸- نمایش یک لیست به همراه List Element های آن

۱-۱۲-کامپوننت های بخش Resources

۱-۱۲-۱-کامپوننت Font

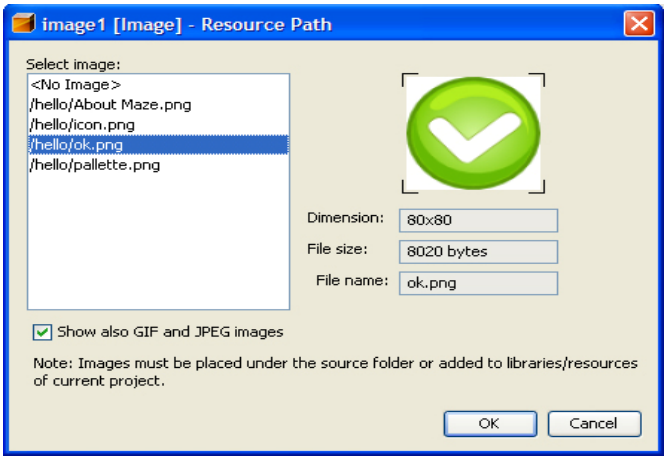
از این کامپوننت برای ایجاد یک نوع فونت دلخواه، باتعیین سایز، نوع وحالت آن به کار می رود. سپس شما می توانید در هر جا که یک متن دارید این فونت را به خاصیت فونت آن متن نسبت دهید. در شکل ۱-۹ شما پنجره تنظیمات فونت را مشاهده می نمایید.



شکل ۱-۹- پنجره تنظیمات فونت

۱-۱۲-۲-کامپوننت Image

با استفاده از این کامپوننت شما می توانید به یک عکس در پروژه خود دسترسی داشته باشید. برای این منظور ابتدا باید عکس مورد نظر خود را با فرمت PNG را در پوشه پروژه در شاخه src/ کپی نمایید. سپس با کلیک بر روی Resource Path، پنجره ای باز شده و شما می توانید آدرس عکس مورد نظر را انتخاب نمایید یا به طور دستی در قسمت Resource Path آدرس آن را بنویسید.



شکل۱-۱۰- انتخاب آدرس عکس برای Resource Path

۱-۱۲-۳-کامپوننت Ticker

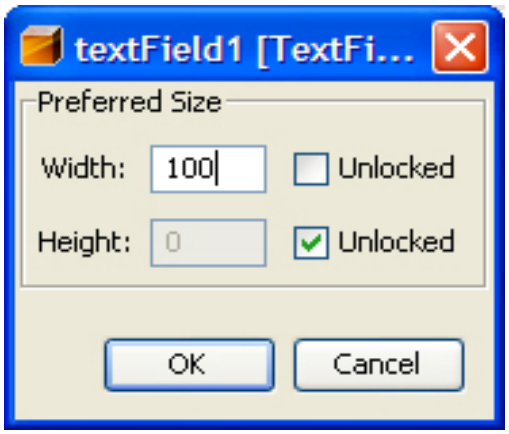
این کامپوننت یک رشته است که در عرض صفحه متناوباً حرکت می کند. سرعت و راستای حرکت بوسیله وسیله (تلفن همراه) کنترل می شود و برنامه کاربردی می تواند حرکت آن را برای جلوگیری از هدر رفتن انرژی، زمانی که کاربر غیر فعال است، در یک مدت زمان مشخص متوقف نماید.

۱-۱۳- تنظیم خاصیت ها در MIDP

۱-۱۳-۱- تنظیم Preferred Size

یک MIDP Item یا یک کامپوننت، یک خاصیت Preferred Size در MIDP 2.0 دارد که قابل تنظیم می باشد. Preferred Size به کل ناحیه مورد نیاز برای کامپوننت اشاره دارد که فضا برای گنجایش کامپوننت، برچسب وابسته و هر فضای اضافی لازم برای آرایش آن را شامل می شود. Preferred Size معمولاً کوچکترین اندازه یک کامپوننت است که می تواند بدون محتوی Clipping و Wrapping text باشد و اگر محتوا تغییر کند ممکن است دوباره محاسبه شود.

شما می توانید Preferred size یک کامپوننت را با یک مقدار عددی مثبت برای عرض و ارتفاع یا هر دو مشخص نمایید. اگر شما در خاصیت Prefereed size گزینه Unlocked را تیک بزنید، سایز کامپوننت به طور خودکار و بسته به اندازه آن تعیین می شود و مثلاً برای کامپوننت TextField اگر این گزینه تیک خورده باشد، کاربر می تواند به تایپ خود ادامه داده و اندازه TextField متناسب با متن تا MaxSize مشخص شده ادامه یابد.



شکل ۱-۱۱- تنظیم خاصیت Prefereed size

۱-۱۳-۲-تنظیم خاصیت Appearance

دو کامپوننت StringItem و ImageItem یک صفت Appearance دارند که می توانند در سازندشان مقدار دهی شود. این صفت می تواند یکی از مقادیر زیر را داشته باشد:

- PLAIN
- HYPERLINK
- BUTTON

این مقادیر را می توانید در خاصیت Appearance کامپوننت مربوطه انتخاب نمایید یا به طور دستی در کد وارد نمایید. برای مثال برای مقداردهی کردن این خاصیت برای یک کامپوننت StringItem شبیه یک کلید، سازنده آن باید به صورت زیر باشد.

```
stringItem1 = new StringItem("Button","Button",Item.BUTTON);
```

توجه : برای ساختن یک آیتم تعاملی به عنوان یک دکمه یا یک هایپر لینک[1]، آن باید یک یا چند دستور داشته باشد و باید CommandListener را برای شنیدن آن دستورات تنظیم نمایید.

۱-۱۳-۳- ایجاد Text Wrapping

برای این که یک جمله از یک مکان مشخص را وادار نماییم که به خط بعد برود، یکی از موارد زیر را انجام می دهیم:

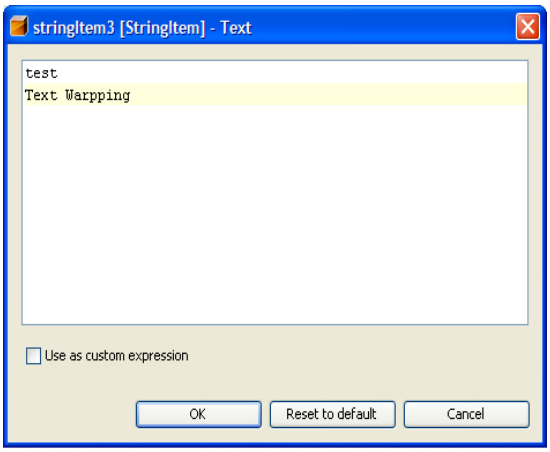
- به طور دستی \n را مستقیماً در محتوی StringItem در سورس کد وارد می نماییم. برای مثال به کد زیر توجه نمایید:

```
stringItem3 = new StringItem("stringItem3", "test\nText Warpping");
```

- با تایپ متن مورد نظر در محیط ویژوال و در خاصیت Text آن.

کاراکتر خط جدید (\n) در یک StringItem تنها یا در StringItem های مجاور، شکست های ردیفی زیادی به تعداد کاراکترهای خط جدید ایجاد می کند.

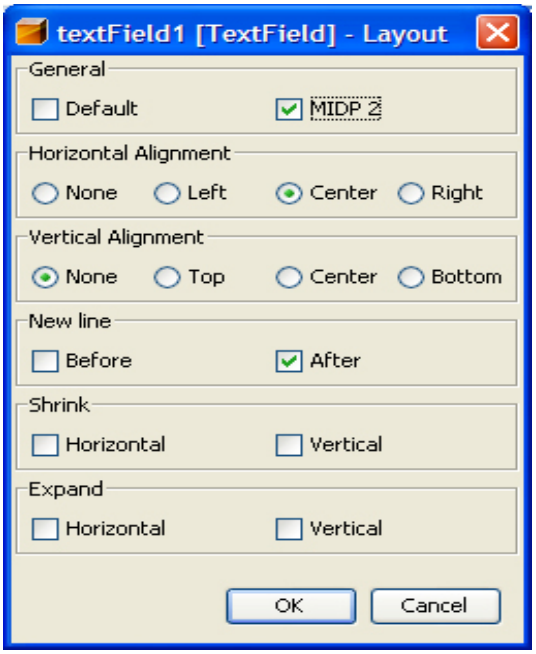
[1] HyperLink



شکل ۱-۱۲- ورود متن در خاصیت Text

۱-۱۳-۴- خاصیت Layout

شما اولین کامپوننتی که به محیط طراحی اضافه می کنید در شروع سطر قرار می گیرد. هر کامپوننتی که بعد از این اضافه شود، در سطر بعدی قرار می گیرد. شما می توانید با استفاده ازخاصیت Layout به تراز بندی کامپوننت ها بر روی صفحه بپردازید. بعضی رفتارهای این خاصیت به طور اتوماتیک، با پیاده سازی وسیله مشخص برای موارد خاص تعیین می گردند. در شکل زیر گزینه های مختلف موجود برای ترازبندی کامپوننت ها را مشاهده می نمایید.



شکل ۱-۱۳- پنجره تنظیمات Layout

۱-۱۴- ایجاد یک تیکر[1] در حالت VMD

تیکر یک خاصیت صفحه است و موقعیتش با وسیله مقصد مشخص می گردد. ایجاد تیکر دو مرحله دارد:

۱- معرفی یک تیکر جدید

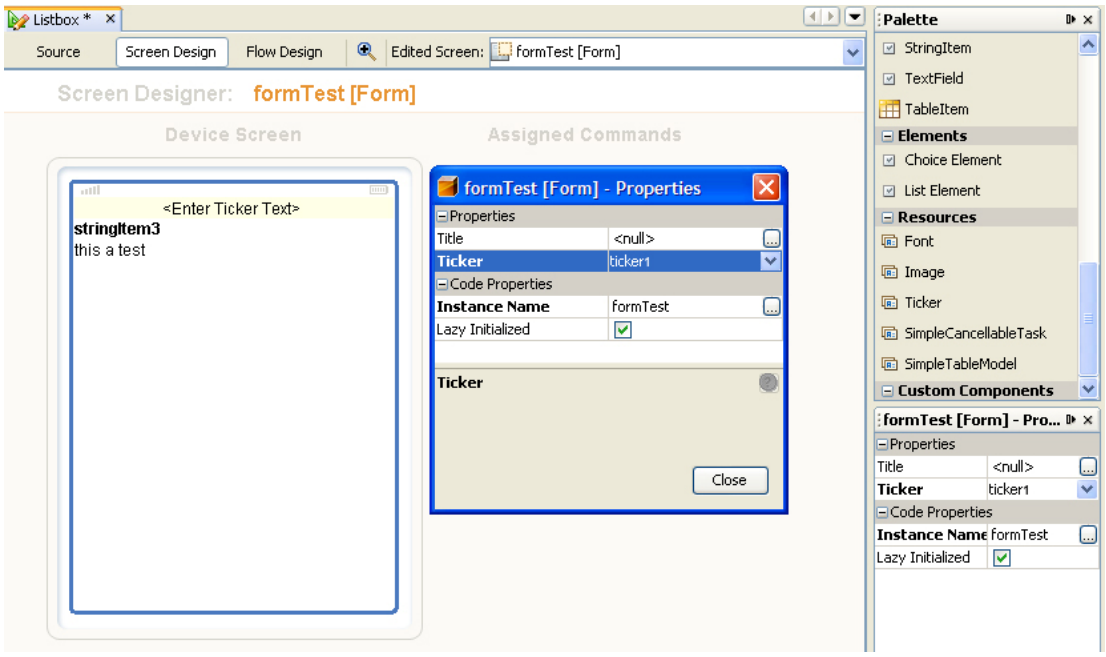
۲-مشخص کردن آن تیکر در خاصیت تیکر کلاس Displayable

مراحل زیر برای ایجاد یک تیکر در طراح UI استفاده می شود:

۱. بر روی کامپوننت تیکر در پالت کلیک نموده وآن را بر روی صفحه قرار می دهیم. کامپوننت تیکر جدید در پنجره Inspector در پوشه Resources قرار می گیرد.
۲. تیکر جدید را انتخاب کرده و در خاصیت String آن متن مورد نظر خود را وارد نمایید.
۳. بر روی صفحه مورد نظر دابل کلیک کرده و Ticker1 را از لیست کشویی آن در خاصیت تیکر انتخاب نمایید.

انجام این مرحله برای کلاس Displayable برای نمایش لازم است و به کلاس می گوید که یک تیکر برای نمایش وجود دارد و مشخص می نماید که باید نمایش داده شود. کد تولید شده برای انجام این عمل به صورت زیر است:

```
This.setTicker(ticker1);
```



شکل ۱-۱۴- ایجاد یک Ticker

[1] Ticker

۱-۱۵- فایل توصیف کننده برنامه جاوا (JAD)

هر فایل MIDlet suite با یک فایل توصیف کننده برنامه جاوا همراه می گردد. این فایل شامل یک مجموعه صفات از قبل تعیین شده می باشد که به نرم افزار مدیریت برنامه های جاوا[1] اجازه می دهد که به شناسایی، بازیابی و نصب MIDlet بپردازد. فایل JAD، نرم افزار مدیریت برنامه کاربردی را قادر می سازد که بر روی یک دستگاه تلفن همراه به شناسایی MIDlet Suite دستگاه، قبل از بارگذاری فایل JAR به طور کامل بپردازد.

در IDE، فایل JAD به طور اتوماتیک هنگامی که شما یک MIDlet Suite را ایجاد می نمایید، ساخته می شود. محتوای فایل JAR را شما می توانید با دابل کلیک بر روی فایل JAR موجود در پنجره فایل مشاهده نمایید و در صورت نیاز آن را اصلاح نمایید. صفات مورد نیاز در یک فایل JAD عبارتند از:

MIDlet-Name: این خاصیت نام MIDlet suite را مشخص می نماید.

MIDlet-Version: شماره نسخه MIDlet suite را مشخص می نماید.

MIDlet-Vendor: سازمانی که MIDlet suite را فراهم می نماید، مشخص می کند.

صفات دیگری برای استفاده در دسترس می باشد که این صفات بستگی به نسخه MIDP ای دارد که توسط پیکربندی پروژه پشتیبانی می شود. این صفات عبارتند از:

MIDlet-JAR-URL: مکان و نام فایل JAR برای این MIDlet suite.

MIDlet-JAR-Size: این صفت اندازه فایل JAR ساخته شده از این MIDlet suite را مشخص می نماید. این خاصیت، یک خاصیت فقط خواندنی می باشد که هر زمان که فایل JAR ساخته شود به روز رسانی می گردد.

MicroEdition-Profile: پیکربندی J2ME مورد نیاز که از همان فرمت و مقدار به عنوان خاصیت سیستم استفاده می نماید. مثلاً MIDP-2.0

MIDlet-Description: توصیف کننده MIDlet suite.

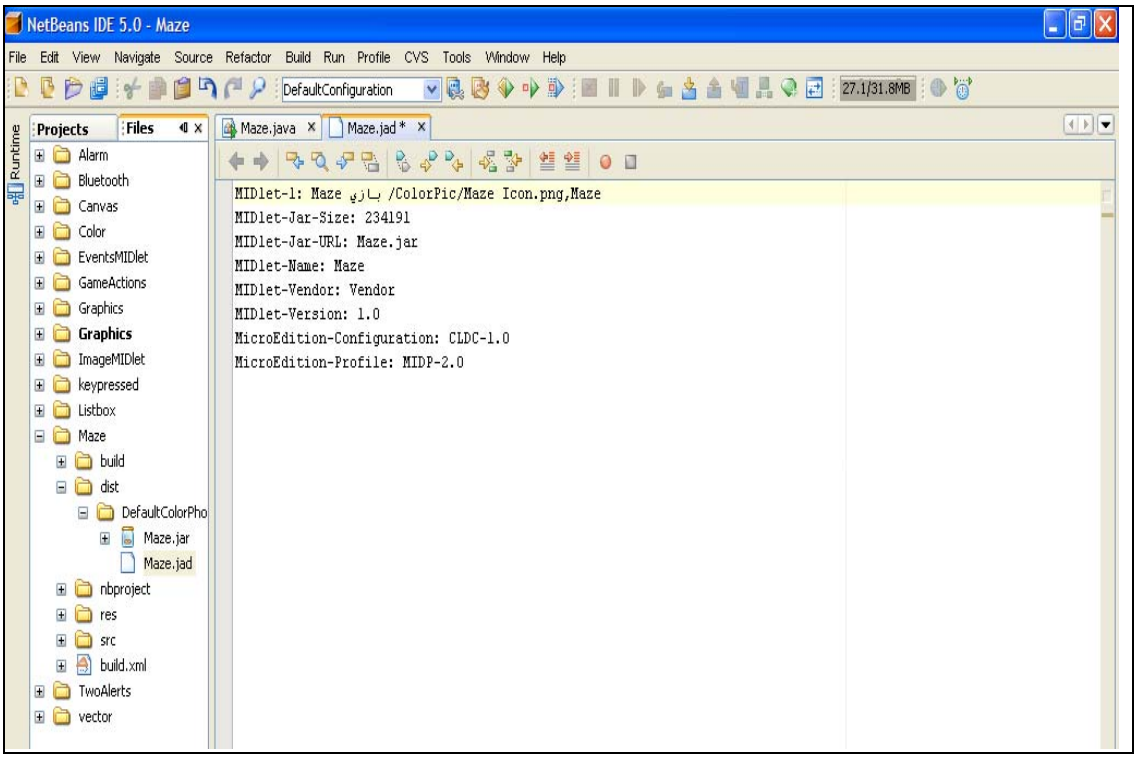
MIDlet-Icon: نام یک فایل PNG که در داخل فایل JAR برای نمایش MIDlet Suite استفاده می شود.

MIDlet-Info-URL: یک URL برای ذخیره اطلاعات بیشتر، که MIDlet suite را توصیف می نماید.

MIDlet-Data-Size: حداقل مقدار بایت داده پایدار که MIDlet نیاز دارد را مشخص می نماید.

در شکل ۱-۱۵ محتویات یک فایل JAD را مشاهده می نمایید.

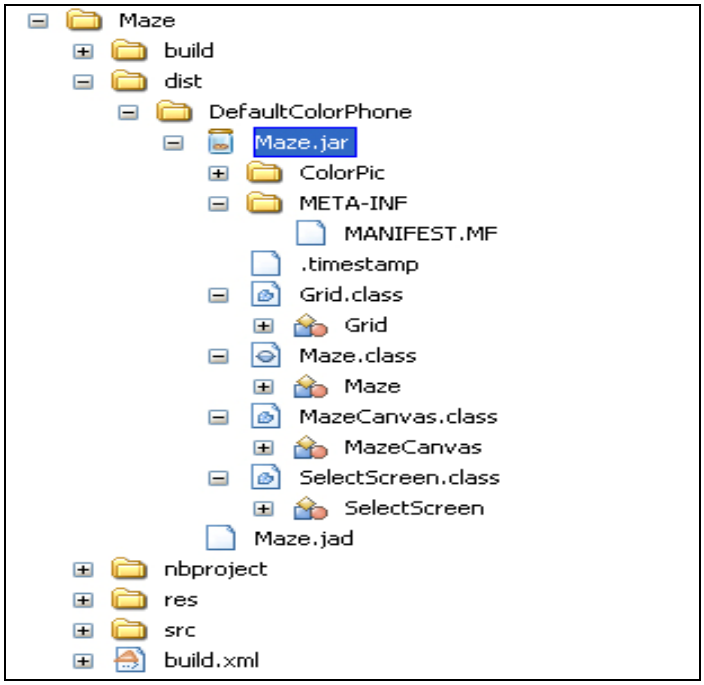
[1] Application Management Software



شکل ۱-۱۵- محتویات فایل JAD

۱-۱۶- محتویات فایل JAR

محتویات فایل JAR کلاس Maze را در شکل زیر مشاهده می نمایید.



شکل۱-۱۶- محتویات فایل JAR کلاس Maze

۱-۱۷- ایجاد یک Canvas

از کلاس Canvas برای ترسیم های گرافیکی در برنامه های کاربردی مثل بازی ها و همچنین برای اداره کردن رویدادهای سطح پایین استفاده می گردد. Canvas کنترل کاملی بر روی آنچه در صفحه کشیده می شود می دهد. با این حال باید همه این اعمال را به طور دستی برنامه نویسی کنید، چون این اعمال نمی تواند در طراح UI انجام شود.

از آنجایی که Canvas با کامپوننت های Screen در تقابل است، یک برنامه کاربردی می تواند از ترکیبی از کامپوننت های Screen و Canvas در برنامه کاربردی استفاده کند. برای مثال شما می توانید از لیست برای فراهم کردن منوی انتخاب یک بازی استفاده نمایید و از Canvas برای ایجاد خود بازی.

برای استفاده از کلاس Canvas باید آن را در یک برنامه کاربردی به صورت زیر کلاس بسازید و متد paint() را در زیر کلاس پیاده سازی نمایید. شما باید یک Displayable جدید که از Canvas، مشتق می شود را ایجاد نمایید. در کد زیر که توسط محیط برنامه نویسی نت بینز تولید شده است، دستورات لازم برای ایجاد یک Canvas را مشاهده می نمایید.

```
import javax.microedition.lcdui.*;

public class MIDPCanvas extends Canvas implements CommandListener {
    /**
     * constructor
     */
    public MIDPCanvas() {
        try {
            // Set up this canvas to listen to command events
            setCommandListener(this);
            // Add the Exit command
            addCommand(new Command("Exit", Command.EXIT, 1));
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * paint
     */
    public void paint(Graphics g) {
        g.drawString("Sample Text",0,0,Graphics.TOP|Graphics.LEFT);
    }

    /**
     * Called when a key is pressed.
```

```
    */
     protected  void keyPressed(int keyCode) {
     }

    /**
     * Called when a key is released.
     */
      protected  void keyReleased(int keyCode) {
      }

    /**
     * Called when a key is repeated (held down).
     */
      protected  void keyRepeated(int keyCode) {
      }

    /**
     * Called when the pointer is dragged.
     */
      protected  void pointerDragged(int x, int y) {
       }

    /**
     * Called when the pointer is pressed.
     */
     protected  void pointerPressed(int x, int y) {
      }

    /**
     * Called when the pointer is released.
     */
    protected  void pointerReleased(int x, int y) {
     }

    /**
     * Called when action should be handled
     */
     public void commandAction(Command command, Displayable displayable) {
      }

}
```

۱-۱۸- معرفی کلاس Graphics

کلاس گرافیک یک کلاس پایه است که برای استفاده از این کلاس باید javax.microedition.lcdui.graphics را به برنامه وارد نمایید. این کلاس برای رسم اشکال دو بعدی هندسی ساده به کار می رود. رسم اشکال ساده ای نظیر دایره، خط، مستطیل، کمان، متن، تصویر. همچنین مستطیل ها و کمان ها را می توان با یک رنگ ساده رنگ آمیزی نمود. همچنین با استفاده از این کلاس می توان به ترسیم مستطیل هایی با لبه گرد پرداخت. این کلاس برای یک مدل رنگ از ۲۴ بیت استفاده می نماید که هشت بیت آن مربوط به رنگ قرمز، هشت بیت دیگر مربوط به رنگ سبز و هشت بیت بقیه مربوط به رنگ آبی می باشد.

این کلاس متدهایی را برای ترسیم اشکال دو بعدی و کار با آن ها را در اختیار کاربر قرار می دهد که عبارتند از:

جدول ۱-۱- لیست متدهای کلاس Graphics

• clipRect • copyArea • drawArc • drawChar • drawChars • drawImage • drawLine • drawRGB • drawRect • drawRegion • drawRoundRect • drawString • drawSubString • fillRect • fillArc • fillRoundRect • fillTriangle • getBlueComponent • getClipWidth	• getClipHeight • getClipX • getClipY • getColor • getTranslateX • getTranslateY • getDisplayColor • getFont • getGrayScale • getGreenComponent • getRedComponent • getStrokeStyle • setClip • setColor • setFont • setGrayScale • setStrokeStyle • translate

حال به معرفی اجمالی این متدها می پردازیم:

متد getColor: این متد وظیفه برگرداندن رنگ جاری را برعهده دارد. خروجی این متد یک مقدار صحیح به شکل 0x00RRGGBB می باشد.

```
Public int getColor()
```

متد getGreenComponent(): این متد مقدار ترکیب رنگ سبز در رنگ جاری را بر می گرداند.

متد getRedComponent(): این متد مقدار ترکیب رنگ قرمز در رنگ جاری را بر می گرداند.

متد getBlueComponent(): این متد مقدار ترکیب رنگ آبی در رنگ جاری را بر می گرداند.

متد setColor: به وسیله این متد شما می توانید رنگ جاری را تغییر دهید.

Public void setColor(int red , int green , int blue)

مقادیر red، green و blue باید بین ۰ تا ۲۵۵ باشد.

Public void setColor(int RGB)

متد getFont: با استفاده از این متد شما می توانید مشخصات فونت جاری سیستم را به دست آورید.

Public Font getFont()

متد setFont: این متد یک سری خصوصیات را برای فونت جاری تعیین می نماید.

Public void setFont(Font font)

متد getStrokeStyle: این متد نوع حالت خط جاری را که شامل دو حالت SOLID و DOTTED می باشد را بر می گرداند.

Public void getStrokeStyle()

متد setStrokeStyle: با استفاده از این متد شما می توانید نوع حالت خط را برای رسم خط، دایره، مستطیل، مستطیل لبه گرد و کمان تعیین نمایید. پارامتر Style می تواند دارای دو حالت SOLID و DOTTED باشد.

Public void setStrokeStyle(int style)

متد drawLine: از این متد برای رسم یک خط ساده استفاده می گردد. این خط بین مختصات دو نقطه (x1,y1) و (x2,y2) با نوع حالت خط جاری رسم می گردد.

Public void drawLine(int x1 , int y1 , int x2 , int y2)

متد fillRect: از این متد برای رسم یک مستطیل توپر با رنگ جاری استفاده می گردد. اگر اندازه طول یا عرض صفر باشد، هیچ چیزی رسم نمی گردد.

Public void fillRect(int x , int y , int width , int height)

متد drawRect: از این متد برای ترسیم خطوط اطراف یک مستطیل با حالت خط و رنگ جاری به کار می رود.

```
Public void drawRect(int x , int y , int width , int height)
```

متد drawRounRect: از این متد برای ترسیم خطوط اطراف یک مستطیل لبه گرد با حالت خط و رنگ جاری به کار می رود.

```
Public void drawRounRect(int x , int y , int width , int height ,  int  arcWidth
                                                                , int arcHeight)
```

متد fillRoundRect: از این متد برای ترسیم یک مستطیل توپر با لبه های گرد و با رنگ جاری استفاده می گردد.

```
Public void fillRounRect(int x , int y , int width , int height ,
                                                int  arcWidth , int arcHeight)
```

متد fillArc: از این متد برای ترسیم یک قسمتی از دایره یا بیضی به صورت توپر با رنگ جاری استفاده می گردد.

```
Public void fillArc(int x , int y , int width , int height ,
                                int startAngle , int endAngle)
```

متد drawArc: از این متد برای ترسیم یک کمان از دایره یا بیضی با حالت خط و رنگ جاری استفاده می گردد. پارامترهای این متد همانند متد fillArc می باشد.

متد drawString: از این متد برای ترسیم یک متن ساده با رنگ و فونت جاری در موقعیت (x,y) و در نقطه لنگر داده شده به کار می گردد. این متد یک سری مقادیر ثابت برای تعیین موقعیت متن توسط نقاط لنگر دارد که در زیر لیست آن ها را مشاهده می نمایید.

- HCENTER
- VCENTER
- TOP
- LEFT
- RIGHT
- BOTTOM
- BASELINE

در زیر دو نمونه از کاربرد این متد را مشاهده می نمایید:

```
g.drawString("Test" , 10, 20 , TOP | LEFT)
g.drawString("Ali", 100 , 42 , BASELINE | HCENTER)
```

متد drawSubString: از این متد برای رسم یک زیر رشته از یک رشته اصلی بر روی صفحه نمایش استفاده می گردد. این متد شبیه به متد drawString است.

```
Public  void drawSubString( String str , int offset , int len , int x ,
                                                    int y , int anchor)
```

متد drawChar: از این متد برای ترسیم یک کارکتر خاص با فونت و رنگ جاری استفاده می گردد.

```
Public void drawChar(Char character , int x , int y , int anchor)
```

متد drawChars: این متد همانند متد drawSubString برای ترسیم یک سری کارکتر خاص پشت سر هم از موقعیت داده شده در آرایه کارکتر ها با فونت و رنگ جاری استفاده می گردد.

```
Public void drawChars(Char[] data , int offset , int len , int x ,
                                                    int y , int anchor)
```

متد translate: این متد وظیفه برگرداندن زمینه گرافیکی مبدا را به نقطه (x,y) در مختصات جاری سیستم را بر عهده دارد. برای مثال فراخوانی translate(1,2) و سپس translate(3,4) نتیجه انتقال را در translate(4,6) قرار می دهد.

دو متد getTranslateX() و getTranslateY(): این دو متد وظیفه برگرداندن مختصات x و y انتقال را در زمینه گرافیکی مبدا بر عهده دارد.

متد fillTriangle: از این متد برای ترسیم یک مثلث توپر با رنگ جاری استفاده می گردد.

```
Public void fillTriangle (int x1 , int y1 , int x2 , int y2 , int x3 , int y3)
```

متد drawImage: از این متد برای ترسیم یک تصویر معین با استفاده از نقاط لنگر استفاده می گردد.

```
Public void drawImage (Image img , int x , int y , int anchor)
```

متد drawRegion: از این متد برای کپی یک ناحیه معین از تصویر مبدا به یک موقعیت در مقصد استفاده می گردد. همچنین اعمال چرخش و بازتاب تصویر با استفاده از توابع تغییر شکل امکان پذیر می باشد.

```
Public void drawRegion( Image src , int x-src , int y-src , int width ,
                                int  height , int transform , int x-dest , int y-dest , int anchor)
```

در زیر برخی از این توابع تغییر شکل که از کلاس Sprint می باشد را مشاهده می نمایید.

- Sprint.TRNS_NONE
- Sprint.TRNS_ROT90
- Sprint.TRNS_ROT180
- Sprint.TRNS_ROT270
- Sprint.TRNS_MIRROR_ROT90
- Sprint.TRNS_MIRROR_ROT180

۱-۱۹-کلاس Display

این کلاس برای نمایش صفحه یا فرم هایی که در برنامه استفاده کردیم، استفاده می شود. برای نشان دادن یک صفحه یا فرم قابل نمایش، ابتدا باید یک شیء از نوع Display تعریف کنیم. سپس با استفاده از متد getDisplay آن را مجاب سازیم و سرانجام با استفاده از متد setcurrent صفحه یا فرم دلخواه را نمایش دهیم.

تعدادی از متدهای مهم این کلاس عبارتند :

isColor : اطلاعاتی درباره رنگ هایی که در وسیله پشتیبانی می شود را برمی گرداند.

numColor : تعداد رنگ هایی که در این وسیله ارائه می شود را برمی گرداند.

SetCurrent : یک فرم یا صفحه را به عنوان پیش فرض نمایش می دهد.

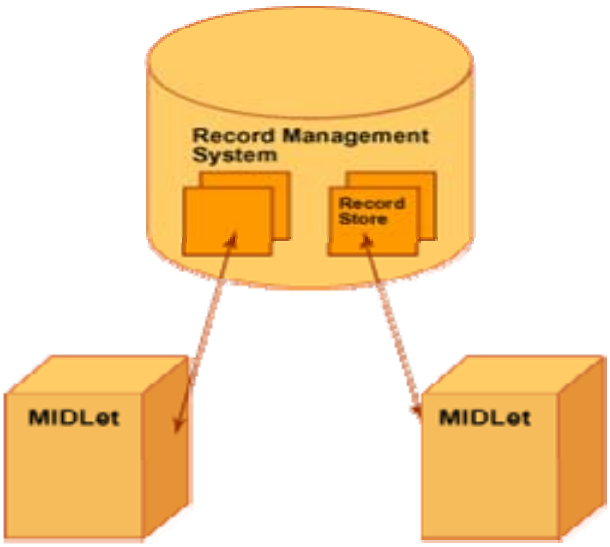
getCurrent : صفحه یا فرم پیش فرض را که در حال نمایش است را برمی گرداند.

فصل دوم – معرفی RMS و نحوه استفاده از آن

فصل دوم – معرفی RMS و نحوه استفاده از آن

۲-۱- معرفی RMS [1]

MIDP یک مکانیزم برای MIDlet به منظور ذخیره سازی پایا داده ها و بازیابی آن فراهم می کند. این مکانیزم یک پایگاه داده ساده است که RMS نامیده می شود. یک پایگاه داده MIDP یا یک رکوردِاستور[2] از مجموعه رکوردهایی تشکیل شده که بعد از خروج از MIDlet، پایا می مانند. وقتی شما MIDlet را دوباره اجرا می کنید، می توانید داده ها را از رکورد استور بازیابی نمایید.



شکل ۲-۱- سیستم مدیریت رکورد

[1] Record Management System
[2] Record Store

۲-۲- رکوردها

یک رکورد، یک داده منفرد است. RMS هیچ محدودیتی روی آنچه که در یک رکورد قرار می دهید، اعمال نمی کند. یک رکورد می تواند شامل یک عدد، یک رشته، یک آرایه، یک تصویر و به عبارت دیگر هر چیزی که دنباله ای از بایت ها است باشد. اگر بتوانید یک دیکدر باینری و انکدر متناسب آن را ایجاد نمایید می توانید در یک رکورد اشیایی با هر اندازه را ذخیره کنید. البته با اعمال محدودیت هایی که سیستم تحمیل می کند.

بسیاری از کسانی که تازه شروع به کار با RMS نموده اند، از اصطلاح رکورد در آن سردر گم می شوند. آن ها می پرسند فیلدها کجا هستند؟ چگونه سیستم رکوردهای منفرد را به ترتیبی از داده های گسسته تقسیم می کند؟ پاسخ ساده است: در RMS یک رکورد هیچ فیلدی ندارد یا به طور دقیق یک رکورد از یک فیلد باینری با اندازه متغیر تشکیل شده است. مسئولیت تفسیر یک رکورد برعهده برنامه کاربردی است.

RMS یک محل ذخیره سازی و یک شناسه منحصر به فرد فراهم می آورد و نه چیز دیگری. در حالی که این تقسیم بندی کار برای یک برنامه کاربردی پیچیده است. با این حال RMS را کوچک و انعطاف پذیر می سازد. یک رکورد استور، یک مجموعه مرتب از رکورد هاست. رکوردها نهادهای مستقلی نیستند بلکه همه باید وابسته به یک رکورد استور باشند و دسترسی های انجام شده به همه رکوردها از طریق رکورد استور انجام می شود. در واقع رکورد استور تضمین می کند که همه رکوردها بدون خرابی احتمالی در داده ها، نوشته و خوانده شوند.

وقتی یک رکورد ایجاد می شود، رکورد استور یک شناسه منحصر به فرد به آن نسبت می دهد، مقداری صحیح که شناسه رکورد[1] نامیده می شود. اولین شناسه رکورد برابر ۱ قرار داده می شود و دومی برابر ۲ و .... حتی اگر رکوردی حذف گردد، باز هم آخرین مقدار منتسب به شناسه رکورد مشخص است و به رکورد جدید مقداری بعد از آخرین انتساب قبلی نسبت داده می شود.

RecordId	Data
1	Data 1 .....
2	Data 2 .....
3	Data 3 .....
5	Data 5 .....
6	Data 6 .....
n	Data n .....

شکل۲-۲- ساختار رکوردها

[1] Record Id

نام ها برای مشخص کردن رکورد استورها در MIDlet suite به کار می روند نام یک رکورد استور از ۱ تا ۳۲ کارکتر یونیکد تشکیل شده است و باید در یک MIDlet suite که رکورد استور را ایجاد کرده، منحصر به فرد باشند. در MIDP 1.0 ، رکورد استورها نمی توانند با MIDlet suite های دیگر به اشتراک گذاشته شوند. MIDP 2.0 به طور اختیاری به یک MIDlet suite اجازه به اشتراک گذاشتن رکورد استور را با سایر suite ها در اختیار می گذارد.

رکورد استور ها همچنین زمان نقش بستن[1]، اطلاعات نسخه را نگهداری می کنند تا برنامه کاربردی بتواند آخرین زمان دستکاری و تغییر آن را مشخص کند.

۲-۳- محدودیتهای ذخیره سازی

مقدار حافظه در دسترس برای ذخیره سازی داده ها، بر مبنای رکورد از یک وسیله به وسیله دیگر متفاوت است.MIDP تشخیص می دهد وسیله ها به صورت رزرو، حداقل ۸ kB حافظه غیر فرار به عنوان حافظه داده پایدار نیاز دارند. مشخصه ها هیچ محدودیتی برای اندازه یک رکورد منفرد مطرح نمی کنند، اما محدویت فضا بین وسیله ها متفاوت است. RMS رکورد هایی را برای تعیین اندازه یک رکورد واحد، مجموع اندازه رکورد استور و مقدار فضای حافظه باقی مانده برای داده ها فراهم می آورد. به خاطر بسپارید که حافظه پایدار، یک حافظه اشتراکی با منابع کوچک و اقتصادی از نظر استفاده است.

هر MIDlet suite که از RMS استفاده می کند، باید حداقل تعداد بایتهای حافظه داده مورد نیاز خود را با تنظیم صفت MIDlet-Data-Size در JAR manifest و توصیف کننده برنامه کاربردی مشخص کند. این مقدار را بزرگتر از حداکثر فضای مورد نیاز قرار ندهید، چرا که ممکن است وسیله از نصب MIDlet suite ای که حافظه داده مورد نیاز آن از فضای موجود تجاوز کند، سرباز زند. اگر این صفت از قلم بیفتد وسیله گمان می کندکه MIDlet suite برای حافظه داده خود به هیچ فضایی نیاز ندارد. در عمل بیشتر وسیله ها به هر برنامه اجازه می دهند تا از حد حافظه مورد نیاز مشخص شده تجاوز کنند، اما وابستگی به این رفتار ایجاد نمی کنند.

توجه: برخی پیاده سازی های MIDP نیاز دارند که صفات بیشتری در ارتباط با نیازمندیهای محل ذخیره سازی انجام دهند.

[1] Time-Stamp

پکیج rms

MIDP یک مجموعه از کلاس ها و رابط ها را در پکیج javax.microedition.rms فراهم می کند که این مجموعه از کلاس ها و رابط ها پکیج rms را تشکیل می دهند.

کلاس رکورد استور

این کلاس برای به وجود آوردن رکورد استور، حذف رکورد استور، اضافه کردن رکورد، حذف رکورد، خواندن رکورد و تغییر رکورد از رکورد استور ها استفاده می شود.

۲-۴- ایجاد یا باز کردن یک رکورد استور

به منظور ایجاد یا باز کردن یک رکورد استور از متد openRecordStore کلاس رکورد استور استفاده می شود. این متد دو پارامتر می گیرد، پارامتر اول نام رکورد استوری است که می خواهیم آن را باز کنیم یا در صورت موجود نبودن آن را ایجاد کنیم. پارامتر دوم یک پارامتر از نوع بولین است که اگر مقدارآن true باشد در صورت موجود نبودن رکورد استور آن را ایجاد می کند.

```
RecordStore rs = RecordStore.openRecordStore("test_rms",true);
```

۲-۵- اضافه کردن رکوردها

به منظور اضافه کردن رکوردها به رکورد استور، از متد addRecord استفاده می شود. این متد سه پارامتر می گیرد، پارامتر اول آرایه ای از بایتهاست که می خواهید آن را در رکورد استور درج کنید. پارامتر دوم آفست شروع ثبت در آرایه (شماره اولین بایت که ثبت باید از آنجا شروع شود) را مشخص می کند، پارامتر سوم تعداد بایتهایی که باید پس از نقطه شروع به رکورد استور منتقل شوند را مشخص می کند. اگر عمل ثبت موفقیت آمیز باشد مقدار شناسه رکورد برگردانده می شود و در غیر این صورت یک استثناء مانند RecordStoreFullException روی خواهد داد.

```
String recordString = "Hello, RMS WORLD";
byte []recordBytes = recordString.getBytes();
intrecId= rs.addRecord(recordBytes,0,recordBytes.length);
```

۲-۶- خواندن رکوردها

برای خواندن یک رکورد باید شناسه رکورد،آن رکورد را بدانیم. سپس با استفاده از متد getRecord می توانیم محتویات آن رکورد را بخوانیم. این متد یک پارامتر به عنوان ورودی می گیرد که برابر با شناسه رکورد است. رکوردی که می خواهیم محتویات آن را بخوانیم، خروجی آن آرایه ای از بایتهاست که محتویات رکورد خوانده شده است.

```
byte [] readRecord = rs.getRecord(recId);
```

۲-۷- به روز رسانی رکوردها

به منظور به روز رسانی رکوردها از متد setRecord استفاده می شود. این متد چهار پارامتر می گیرد. پارامتر اول، شناسه رکوردی است که می خواهیم محتویات آن را تغییر دهیم و بقیه پارامترهای آن همانند متد addRecord است.

```
recordString = "new data";
recordBytes = recordString.getBytes();
rs.setRecord(recId,recordBytes,0,recordBytes.length);
```

۲-۸- حذف یک رکورد

با داشتن شناسه یک رکورد و با استفاده از متد deleteRecord می توان آن را حذف کرد. این متد یک پارامتر به عنوان ورودی می گیرد که برابر با شناسه رکوردی است که می خواهیم آن را حذف کنیم.

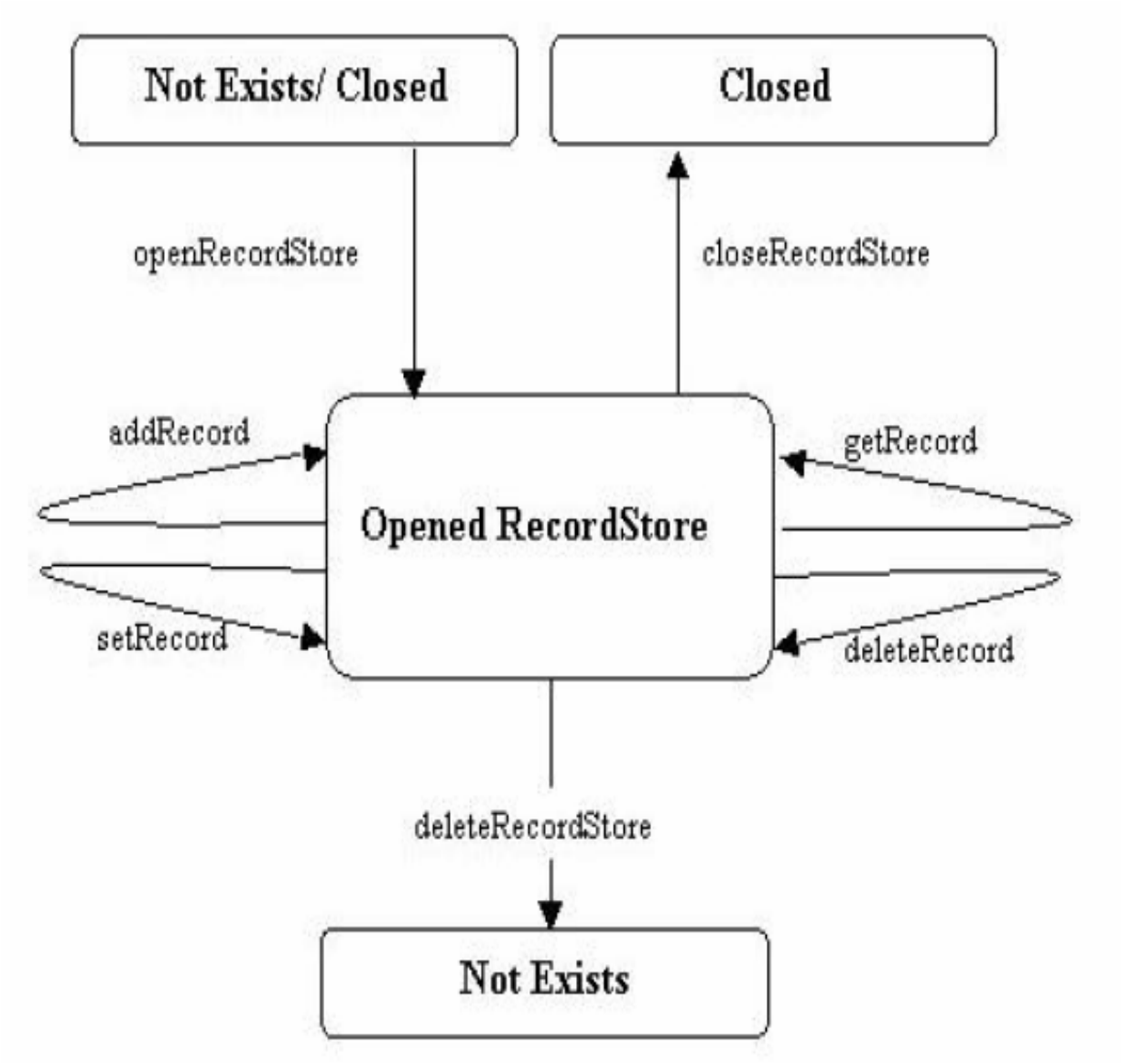
```
rs.deleteRecord(recId);
```

۲-۹- بستن یک رکورد استور

بعد از انجام عملیات مورد نظر مانند حذف رکورد، خواندن رکورد و غیره می توان رکورد استور را با استفاده از متد closeRecordStore بست.

```
rs.closeRecordStore();
```

شکل زیر حالت های مختلف یک رکورد استور را نشان می دهد.



شکل ۲-۳- حالتهای مختلف یک رکورد استور

۲-۱۰- استثناء های RMS

یک سری از استثناءها که در هنگام کار با رکورد استورها ممکن است روی دهند عبارتند از:

۱- InvalidRecordIDException: این استثناء زمانی روی می دهد که شناسه رکورد استفاده شده معتبر نباشد.

۲- RecordStoreFullException: زمانی که عملیات به دلیل کافی نبودن فضای موجود در رکورد استور نمی تواند کامل شود، رخ می دهد.

۳- RecordStoreNotFoundException: زمانی روی می دهد که برنامه سعی در باز کردن رکورد استوری کند که ایجاد نشده است.

۴-RecordStoreNotOpenException: زمانی روی می دهد که سعی در بستن یک رکورد استور را داریم، در حالی که آن باز نیست.

رابط RecordComparator

از این رابط می توان برای مقایسه دو رکورد از نظر این که با هم برابر هستند و یا چه ارتباطی با هم دارند (رکورد اول بزرگتر از رکورد دوم است یا بر عکس) استفاده می شود. به این ترتیب می توان به نظم و ترتیب رکوردها پرداخت.

رابط RecordFilter

با استفاده از این رابط می توان با تعیین یک فیلتر، برای تعیین این که آیا رکوردها با فیلتر گذاشته شده برابر هستند یا خیر استفاده کرد. (فیلتر گذاشته شده بر اساس کاربرد مورد نظر تعیین می شود).

رابط RecordListener

یک رابط شنونده برای رویدادهای اضافه کردن رکورد، حذف رکورد، تغییر رکورد از رکورد استور است.

۲-۱۱- مرتب کردن و جستجوی رکوردها

برای مرتب کردن و جستجوی رکوردها می توان از رابط RecordEnumration استفاده کرد. این رابط بین رکوردها حرکت می کند و می توان به این طریق محتویات آن رکورد یا شناسه آن را به دست آورد و یا با استفاده از رابط های RecordFilter و RecordComparator به مرتب کردن وجستجوی رکوردها پرداخت. این رابط سه پارامتر به عنوان ورودی می گیرد.

۱-RecordFilter: اگر این پارامتر برابر null نباشد برای تعیین این که کدام زیر مجموعه از مجموعه رکوردهای رکورد استور استفاده شوند، استفاده می شود.

۲-RecordComparator: اگر این پارامتر برابر null نباشد به منظور تعیین ترتیب رکوردها در رکورد استور استفاده می شود.

۳- پارامتر سوم از نوع بولین است که اگر true باشد شمارشگر شماره فعلی را به همراه هر گونه تغییرات در رکوردهای رکورد استور نگه می دارد.

۲-۱۲- خواندن رکوردها از RecordEnumration

برای خواندن رکورد بعدی از رکورد استور می توان از متد nextRecord این رابط استفاده کرد این متد یک کپی از محتویات رکورد را بر می گرداند. برای خواندن رکورد قبلی نیز می توان از متد previousRecord استفاده کرد.

```
Recordnumeration re = rs.enumerateRecords(null,null,true);
byte [] readRecord = re.nextRecord ();
```

کد زیر به منظور مرتب کردن رکوردها استفاده شده است و کاربرد مطالب گفته شده را نشان می دهد.

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

public class SortExample extends MIDlet implements CommandListener
{
   private Display display;
   private Alert alert;
   private Form form;
   private Command exit;
   private Command start;
   private RecordStore recordstore = null;
   private Recordnumeration recordnumeration = null;
   private Comparator comparator = null;
   public SortExample ()
   {
     display = Display.getDisplay(this);
     exit = new Command("Exit", Command.SCREEN, 1);
     start = new Command("Start", Command.SCREEN, 1);
     form = new Form("Mixed Recordnumeration", null);
     form.addCommand(exit);
```

```
    form.addCommand(start);
    form.setCommandListener(this);
  }
  public void startApp()
  {
    display.setCurrent(form);
  }
  public void pauseApp()
   {
   }
  public void destroyApp( boolean unconditional )
   {
   }
 public void commandAction(Command command, Displayable displayable)
  {
    if (command == exit)
    {
      destroyApp(true);
      notifyDestroyed();
    }
    else if (command == start)
    {
      try
      {
        recordstore = RecordStore.openRecordStore(
                      "myRecordStore", true );
      }
      catch (Exception error)
      {
        alert = new Alert("Error Creating",
             error.toString(), null, AlertType.WARNING);
        alert.setTimeout(Alert.FOREVER);
        display.setCurrent(alert);
      }
      try
      {
        String outputData[] = {"Mary", "Bob", "Adam"};
        for (int x = 0; x < 3; x++)
        {
          byte[] byteOutputData = outputData[x].getBytes();
          recordstore.addRecord(byteOutputData, 0,
                    byteOutputData.length);
        }
      }
      catch ( Exception error)
      {
        alert = new Alert("Error Writing",
                error.toString(), null, AlertType.WARNING);
        alert.setTimeout(Alert.FOREVER);
        display.setCurrent(alert);
```

```
}
try
{
  StringBuffer buffer = new StringBuffer();
  Comparator comparator = new Comparator();
  recordnumeration = recordstore.enumerateRecords(
                     null, comparator, false);
  while (recordnumeration.hasNextElement())
  {
   buffer.append(new String(recordnumeration.nextRecord ()));
   buffer.append("\n");
  }
  alert = new Alert("Reading", buffer.toString() ,
           null, AlertType.WARNING);
  alert.setTimeout(Alert.FOREVER);
   display.setCurrent(alert);
}
catch (Exception error)
{
   alert = new Alert("Error Reading",
            error.toString(), null, AlertType.WARNING);
   alert.setTimeout(Alert.FOREVER);
   display.setCurrent(alert);
}
try
{
  recordstore.closeRecordStore();
}
catch (Exception error)
{
   alert = new Alert("Error Closing",
            error.toString(), null, AlertType.WARNING);
   alert.setTimeout(Alert.FOREVER);
   display.setCurrent(alert);
}
if (RecordStore.listRecordStores() != null)
{
  try
  {
     RecordStore.deleteRecordStore("myRecordStore");
     recordnumeration.destroy();
  }
  catch (Exception error)
  {
    alert = new Alert("Error Removing",
          error.toString(), null, AlertType.WARNING);
    alert.setTimeout(Alert.FOREVER);
    display.setCurrent(alert);
  }
}
```

```
      }
    }
}
class Comparator implements RecordComparator
{
  public int compare(byte[] record1, byte[] record2)
  {
     String string1 = new String(record1),
               string2= new String(record2);
     int comparison = string1.compareTo(string2);
     if (comparison == 0)
       return RecordComparator.EQUIVALENT;
     else if (comparison < 0)
       return RecordComparator.PRECEDES;
     else
       return RecordComparator.FOLLOWS;
  }
```

۲-۱۳- توابع مفید دیگر

getNumRecord: تعداد رکوردهای یک رکورد استور را بر می گرداند.

getname: نام یک رکورد استور را بر می گرداند.

ListRecordStore: این متد یک آرایه از نام تمام رکورد استور ها را بر می گرداند.

getRecordSize: سایز یک رکورد را بر حسب بایت بر می گرداند.

getsize: مقدار فضایی را که مجموع رکوردهای یک رکورد استور گرفته اند را نشان می دهد.

getsizeAvailable: مقدار فضای خالی موجود در رکورد استور را که رکوردها می توانند به آن اضافه شوند را نشان می دهد.

منابع و مراجع

1. Martin de Jode, Jonathan Allin, Darren Holland, Alan Newman and Colin Turfus, ***Programming Java 2 Micro Edition on Symbian OS*** , Copyright @ 2004 Symbian

2. James Keogh, ***J2ME: The Complete Reference***

3. Michael Juntao Yuan***, Enterprise J2ME: Developing Mobile Java Applications***

4. http://www.symbian.com

5. http://www.netbeans.org

6. http://www.symbiandevnet.com

7. http://java.sun.com/j2me

8. http://www.forum.nokia.com

9. http://www.uiq.com/developer

10. http://www.eclipse.org

11. http://www.borland.com/products/downloads/download jbuilder.html

12. http://www.symbian.com/phones

13. http://www.bluetooth.com

14. http://java.sun.com/products/j2mewtoolkit/download-2 1.html

15. http://www.sun.com

16. http://developer.java.sun.com/developer/onlineTraining/Programming/JDCBook/index.htm

17. http://www.jackwind.net

18. http://www.javasoft.com

19. http://www.Developer.com

20. http://bdn.borland.com

21. http://www.sonyericsson.com

22. http://www.Nokia.com

23. http://developer.java.sun.com/developer/J2METechTips/2001/tt0220.html

24. http://developer.java.sun.com/developer/J2METechTips/2001/tt0622.html

25. http://www.coolarchive.com/icons.php

26. http://www.symbian.com/developer/sdks_series60.asp

27. http://www.smartphone.net

28. http://www.plamsource.com/includes/midp/articles/databasemap

29. http://www.microjava.com/articles/techtalk/rms

30. http://www.java.about.com/od/mobilej2meprogramming/