الفصل الثالث

إضافة الأزرار (commands) إلى البرنامج في J2ME، كيفية بناء التطبيق مع استخدام Commans

بعد أن تعرفنا على كيفية بناء التطبيق الأول وكيفية تحويل هذا الكود إلى تطبيق يعمل على الجوال سنقوم الآن بشرح لكيفية إضافة الأزار والتعامل معها بعد هذا الشرح يفترض بك أن تكون قادر على بناء تطبيق صغير من تلقاء نفسك يجب عليك أولا إنشاء زر جديد ، ثم إضافة هذا الزر إلى الواجهة التي تريد عرضها ، بعد ذلك ستضع الكود الذي تود تنفيذه عند الضغط على هذا الزر.

١ - تعريف الأزرار في البرنامج:

أولاً: سنقوم بتعريف كائن جديد من نوع Command

CODE

```
Command command;
```

٢ - ثم نقوم بإنشاء هذا الكائن باستخدام الأمر new ونقوم بإعطاء الزر بعض الخصائص مثل الاسم الذي سيظهر للمستخدم وكذلك نوع هذا الزر وأولوية الظهور

CODE

```
command = new Command(String label, int
commandType, int priority)
```

ويمكن اختصار الطريقتين السابقتين على الشكل:

CODE

```
Command command = new Command(String
label, int commandType, int priority)
```

أنشأنا كائن من نوع زر ثم أعطيناه بعض الخصائص.

٣ - ربط هذا الزر مع واجهة محددة بعد ذلك نقوم بإضافة هذا الزر إلى الواجهة UI التي نريد عرض الزر معها باستخدام الطريقة:

CODE

```
UI.addCommand(command);
```

مثلاً لو كانت الواجهة TextBox box سنكتب الأمر على الشكل:

CODE

```
box.addCommand(command);
```

٤ - إضافة الكود الذي سينفذ عند الضغط على الزر من أجل إضافة كود لزر أمر معين يجب علينا استخدام implements Interface .

أي إننا سنضمن الواجهة CommandListener في الكلاس الذي سنستخدم فيه الأزرار بهذا الشكل.

CODE

```
public class HelloMidlet extends MIDlet implements
CommandListener  {
```

يوجد في الواجهة CommandListener الطريقة المجردة commandAction(Command c, Displayable s) لهذا يجب تضمين هذه الطريقة في تطبيقنا لأننا استخدمنا implements CommandListener .

تعيد لنا هذه الواجهة كائنان الأول Command c وهو الزر الذي قمنا بضغطه الثاني Displayable s وهي الواجهة UI الذي تحوي هذا الزر تعمل هذه الواجهة على تشغيل المتنصت Listener لكي يتصنت على كبسة الزر يجب أولاً توجيه هذا المتنصت إلى UI محددة من خلال الأمر setCommandListener أي إننا سنخبر المتنصت أن ينفذ الكود عند حدوث كبسة زر ما من تلك الواجهة UI .

والآن سنضع هذا المثال المطور عن مثالنا السابق ونعود بعد ذلك لمناقشة الأشياء الجديدة فيه.

CODE

```
import javax.microedition.midlet.*;

import javax.microedition.lcdui.*;


public class HelloMidlet extends MIDlet implements
CommandListener
{

private Display display;
Command command = new Command("Exit",
Command.EXIT, 0);

TextBox box;

public void HelloMidlet(){}

public void startApp(){


display =Display.getDisplay(this);

box =new TextBox("Simple Example ","Hello
",20,0);
box.addCommand(command);
box.setCommandListener(this);
display.setCurrent(box);
}
```

```
public void pauseApp(){ }

public void destroyApp(boolean unconditional){}

/************* implements CommandListener
****************/

public void commandAction(Command c,
Displayable s)
 {
   if ( c == command && s == box )
     {
      destroyApp(false);
      notifyDestroyed();
     }
 }

}
```

CODE

```
public class HelloMidlet extends MIDlet implements
CommandListener
```

implements : يعني أن هذا الكلاس سيستخدم واجهة أو أكثر من واجهة تفصل بينهما فاصلة صغيرة، Interface1,Interface2 .

CommandListener : عبارة عن واجهة تستخدم لإضافة رد فعل معين تجاه كبسة زر معين.

CODE

```
Command command = new Command("Exit",
Command.EXIT, 0);
```

قمنا بإنشاء الكائن command والذي سيظهر للمستخدم بالاسم Exit.

CODE

```
box =new TextBox("Simple Example ","Hello
",20,0);
```

إنشاء كائن من نوع صندوق نص TextBox عنوان هذا الصندوق Simple Example .

وعدد الحروف التي يسمح بها ٢٠ حرف، أما القيمة ٠ فتمثل TextField.ANY وهذا يعني أن صندوق النص سيتقبل أي حرف أو رقم.

CODE

```
box.addCommand(command);
```

قمنا بإضافة الزر الذي أنشأناه سابقاً إلى box باستخدام الطريقة addCommand الموجودة في جميع الواجهات UI.

CODE

```
box.setCommandListener(this);
```

أخبرنا المتنصت بأن يقوم بالتنصت على الواجهة box باستخدام الأمر setCommandListener الموجود في جميع الواجهات UI أما الكلمة this فتعني الكلاس الحالي الذي نحن فيه، وهذا يعني أننا سوف نقوم بمعالجة كبسة الزر في هذا الكلاس.

CODE

```
public void commandAction(Command c,
Displayable s)
```

هذه هي الطريقة التي تعمل مع الواجهة CommandListener وتعيد بارامتران الأول هو الزر الذي تم كبسه والثاني الواجهة UI التي تحوي هذا الزر.

CODE

```
if ( c == command && s == box )
```

التحقق من أن الزر الذي تم كبسه والتأكد من الواجهة التي تحوي هذا الزر.

CODE

```
    destroyApp(false);
    notifyDestroyed();
```

هذان الأمران يقومان بإغلاق التطبيق. الأول يقوم بتشغيل عملية الإغلاق باستدعاء الدالة destroyApp والثاني يقوم بالتحسس لعملية الإغلاق.