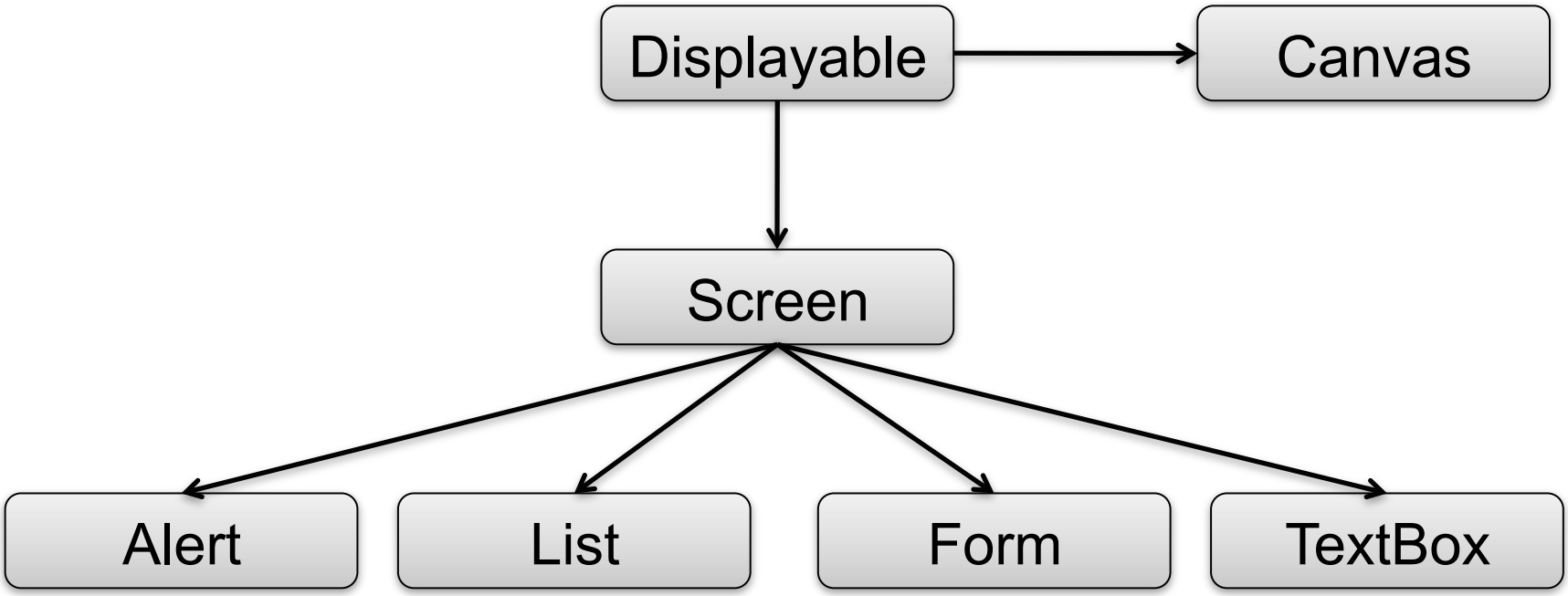
Africa Information
Technology Initiative

Lecture 19:
J2ME UI Elements

AITI 2009

The Displayable Hierarchy



- The appearance of the **`Screen`** sub-classes are device-dependent
- All these classes are defined in javax.microedition.lcdui

Textbox

- Allows the user to enter a String (zipcode,name,password)
- depending on input may be a tedious process

```
public TextBox(String title,
  String text, int maxSize,
  int constraints)
```

  - title: screen title
  - text: initial text on screen
  - maxSize: maximum size of text box
  - constraints: restrict input

TextBox - Constraints

- Constrain the input characters
  - TextField.ANY – allows any type of input supported by the device
  - TextField.NUMERIC– restricts to only integers
  - TextField.DECIMAL– allows numbers with fractional parts
  - TextField.PHONENUMBER – requires a telephone number
  - TextField.EMAILADDR – requires an email address
  - TextField.URL – requires a web address

TextBox - Flags

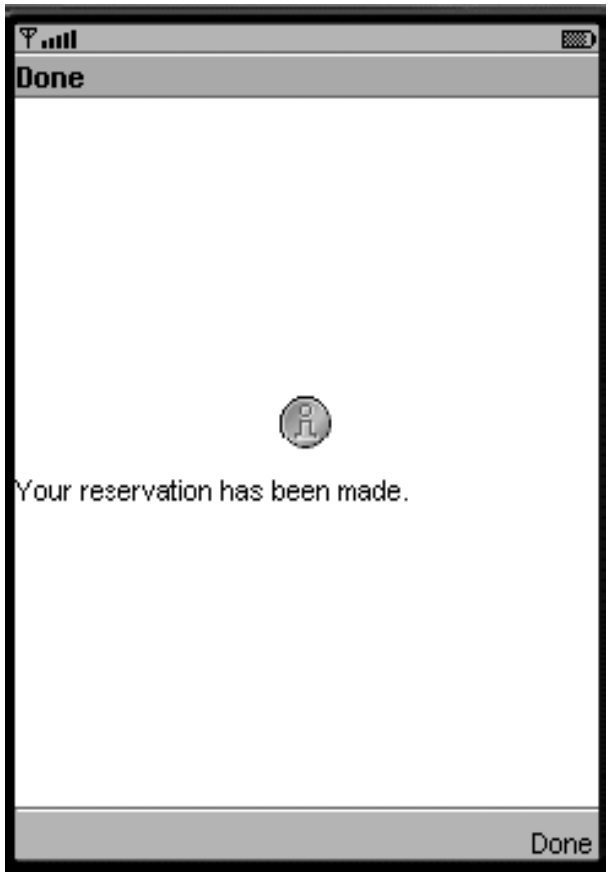
- Flags define behaviour and look of text box
  - TextField.PASSWORD
    - Don’t show user input / Don't save in T9
  - TextField.UNEDITABLE
    - User cannot edit
  - TextField.SENSITIVE
    - Don't save in T9
  - TextField.NON_PREDICTIVE
    - Turn of T9 if on
  - TextField.INITIAL_CAPS_WORD
  - TextField.INTIAL_CAPS_SENTENCE

TextBox - Flags

- Combine flags and constraints with | (bitwise-or)

```
TextBox passwordBox =
    new TextBox("Password", 64,
            TextField.ANY | TextField.PASSWORD);
```

Alerts



- Informative message displayed to user
- Alerts have different types
  - Affects the icon
- An alert can either be timed or user-dismissed

Alerts Types

- Types:
  - ALARM
  - CONFIRMATION
  - ERROR
  - INFO
  - WARNING

- Specify the type that describes your alert

Alert Behavior

- Timed
  - Display for a certain amount of time
  - Ex: “Your transaction complete”
- Modal
  - Displayed until user dismisses it
  - `alert.setTimeout(Alert.Forever);`
  - Ex: “Are you sure you want to quit?”
  - Ex: “Exit without saving?”

Alerts

```
public Alert() or

public Alert(String title, String alertText,
  Image alertImg, AlertType alertType)
```

- Any or all parameters can be null
  - Make sure you set title and alertText before displaying
- Default timeout , but can change timeout length
  - `alert.setTimeout(5000); // 5 second timeout`
- Forever timeout means that it is modal
  - `alert.setTimeout(Alert.FOREVER); //MODAL`

Alerts and Commands

- Of course, you can add commands
- If you don't specify a command for an alert, a command will be added for you
  - Example: "Done"
  - Command object: `Alert.DISMISS_COMMAND`
- As soon as you add a command, the predefined command is removed

Alert Default Behavior

- If you display an Alert, and do not handle any Commands for the Alert
  - By default the previous Displayable will be displayed after the Alert is dismissed
  - Or you can provide the next Displayable to display with another version of setCurrent():

```
display.setCurrent(alert, nextDisplayable);
```

Alerts and CommandListener

- If you want Commands on the Alert to do something special
  - Add Commands to Alert
    - Ex: `alert.addCommand(EXIT_CMD);`
  - Register the listener for the Commands
    - Ex: `alert.setCommandListener(this);`
  - Catch the Commands in the listener

```
public void commandAction(Command c,
                          Displayable d) {
  if (c == EXIT_CMD && d == alert)
     //do something
```

Lists

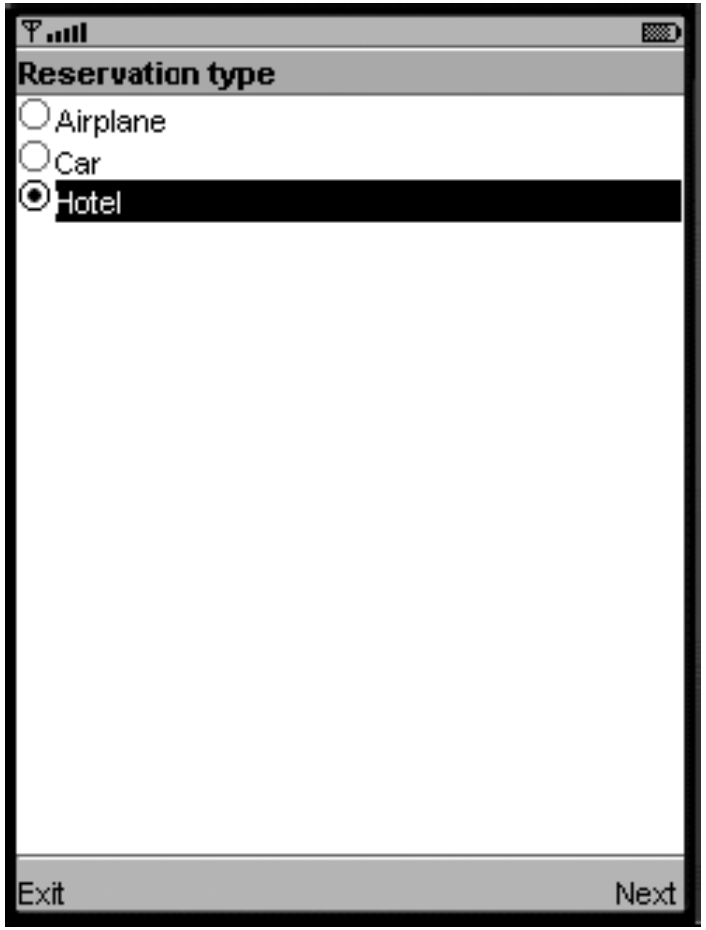
- Users select items (called elements) from choices
- Types:
  - Exclusive, Implicit: Single Selection
    - ex. radio buttons
  - Multiple: Multiple Selections
    - Ex. check list, symptom list
- String or image used to represent each element
- No commands provided for Multiple and Exclusive

Lists and the Users

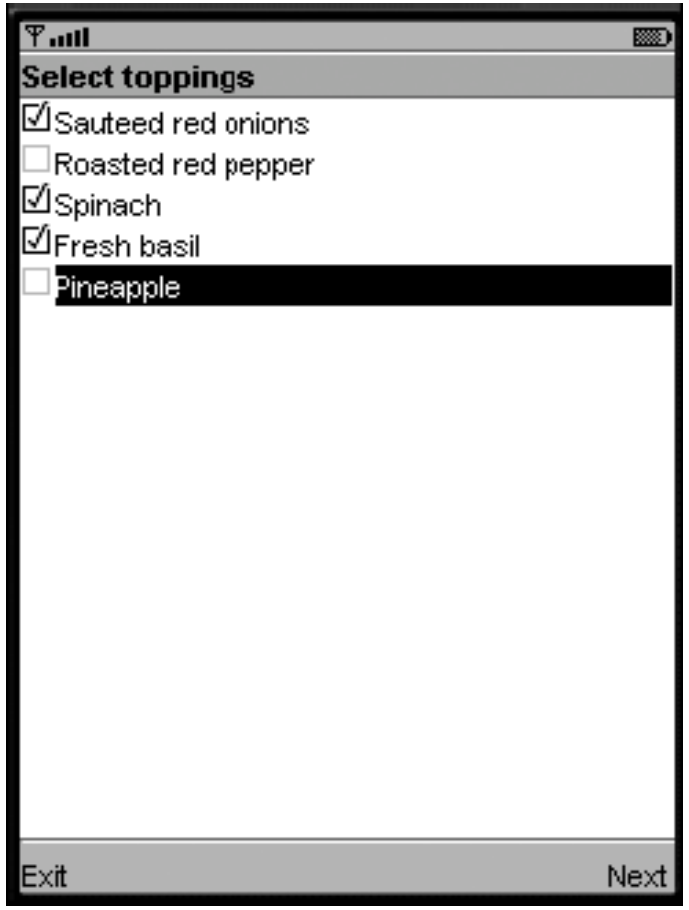
- For Multiple and Exclusive lists, the user:
  - Navigates to element
  - selects element with SELECT (OK) Button
    - Above steps can be repeated for Multiple List
  - Presses Command to signify done

- For Implicit lists, the user:
  - Navigate to element
  - Press OK (Select) Button or Command, to signify done

Exclusive and Multiple Lists

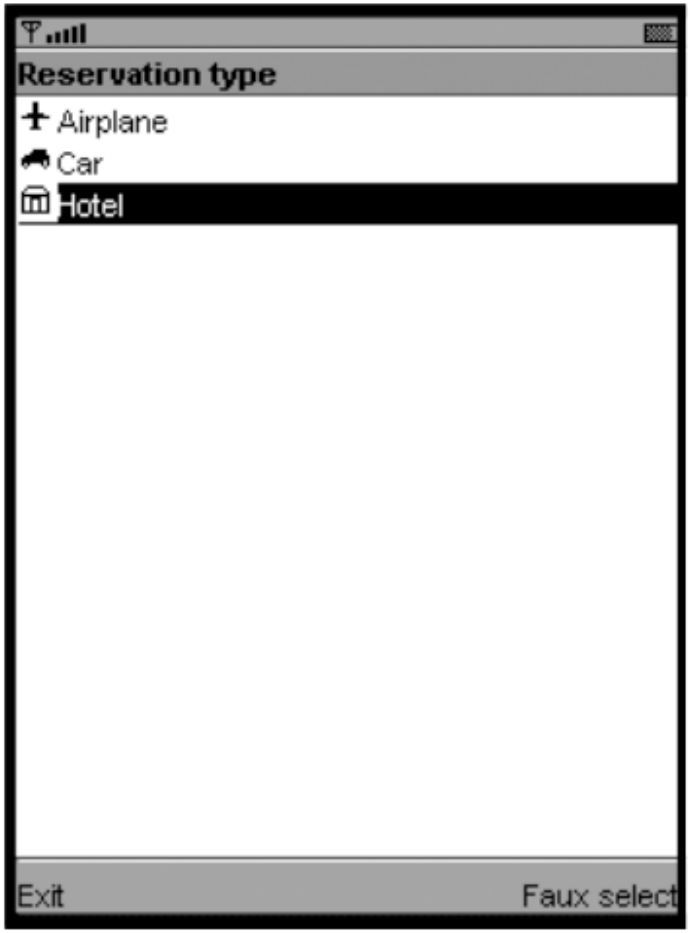


Exclusive



Multiple

Implicit List (with Images)



Creating Lists

```
public List(String title, int type);

public List(String title, int type,
  String[] stringElements,
  Image[] imageElements);
```

- Type:
  - `List.IMPLICIT`, `List.EXCLUSIVE`, or `List.MULTIPLE`
- Image[] can be null (no images)
  - Look at Apress book for info on Images

Event Handling and Implicit Lists

- For Implicit lists, a special command is created that refers to the Select (OK) key
  - `List.SELECT_COMMAND`

```
public void commandAction(Command c, Displayable s) {
  if (c == List.SELECT_COMMAND && s == implicitList)
    // ...
}
```

- You can also add you own commands to Implicits

Multiple and Exclusive Lists

- You can add Commands lists just like Alerts and TextBoxes
- Don't forget to register the listener for the List
- Once a Command is pressed, you can get the List elements that are selected

Determining What is Selected

- Multiple Lists
  - `public boolean isSelected(int index)`
- Exclusive and Implicit
  - `public int getSelectedIndex()`

List Methods

- `public String getString(int elementNum)`
- `public String getImage(int elementNum)`
- `public void setSelectedIndex(int index, boolean selected)`
  - If you want to preselect some elements for the user

Modifying Lists

- `public void set(int elementNum,`
  `String stringPart, Image imagePart)`
- `public void insert(int elementNum,`
  `String stringPart, Image imagePart)`
- `public int append(String stringPart,`
  `Image imagePart)`
- `public void deleteAll()`

Boolean Masks

- `public int getSelectedFlags(boolean[] selectedArray_return)`
  - Returns the number of selected elements in the list
  - Also, for the boolean array, set the value at index to true if the element in the list at index is selected

- `public void setSelectedFlags(boolean[] selectedArray)`
  - Set the elements of the list to selected if the corresponding index in array is true

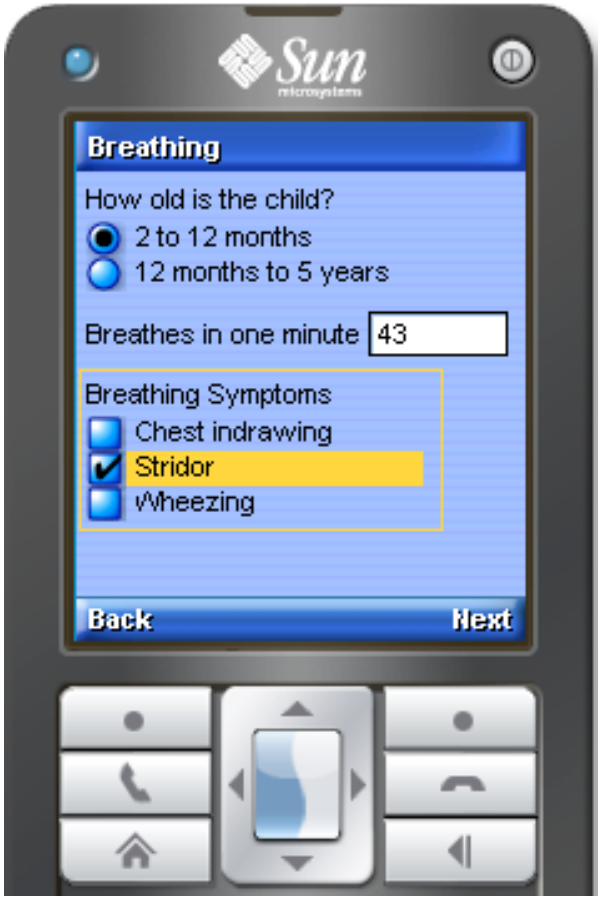
List Example

Forms

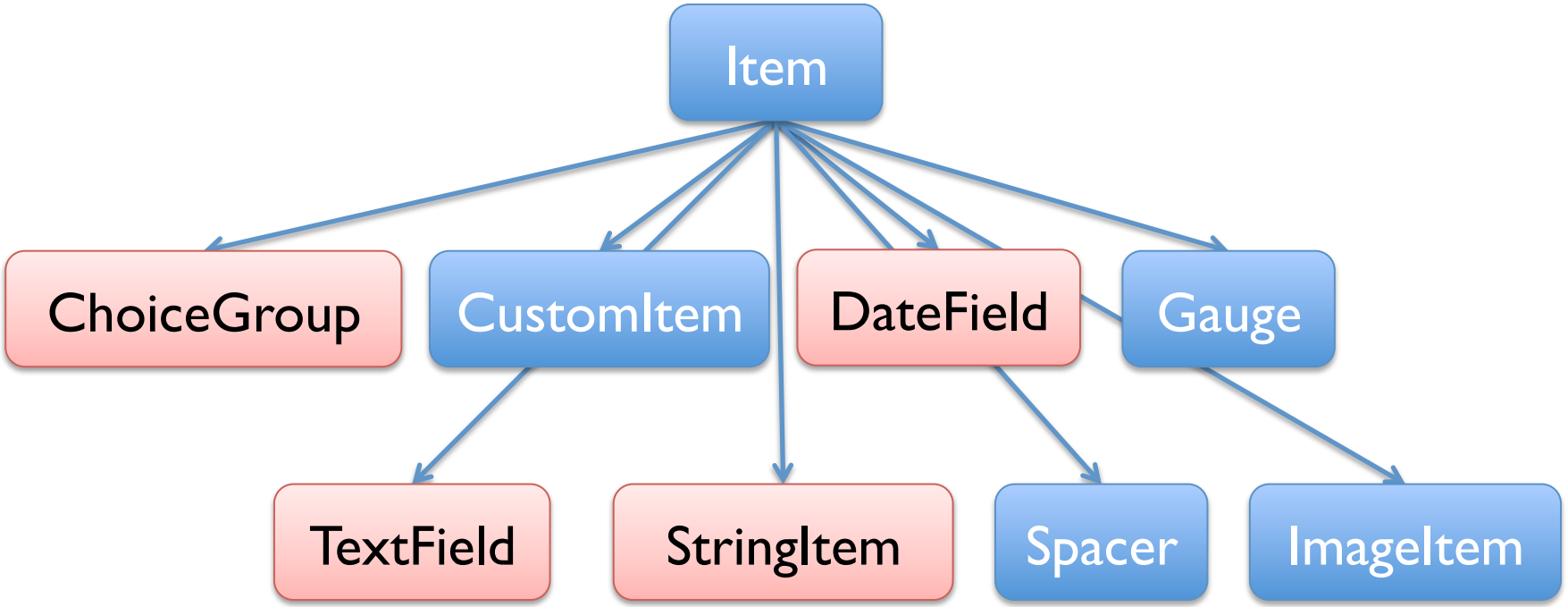
- A form includes collection of UI controls called *Items* on one Displayable

Examples:

- Enter city and country for weather search
- Enter your information
  - Name
  - Address
  - Birthday
- Display text and a picture



Items



- Item defines:
  - `void setLabel(String)` and `String getLabel()`

StringItem

- A simple (labeled) piece of text

```
Form form = new Form("Form Title");
StringItem stringItem =
          new StringItem("Label: ", "text");
form.append(stringItem);
```

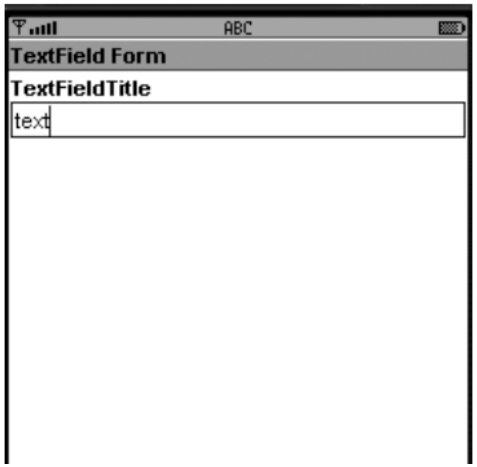
Methods

- `String getText(), String getLabel()`
- `void setText(String), void setLabel(String)`

TextField

- Editable field with a label
- Similar to TextBoxes

```
public TextField(String label,
      String text, int maxSize,
      int constraints)
```



TextField Constraints

- TextFields can limit input. The following constants are defined:
  - ANY allows any type of input
  - NUMERIC restricts the input to numbers
  - DECIMAL allows numbers with decimal values
  - PHONENUMBER requires a telephone number
  - EMAILADDR requires an e-mail address
  - URL input must be a URL
- Flags: PASSWORD,SENSITIVE,UNEDITABLE, NON_PREDICTIVE, INITIAL_CAPS_WORD, and INITIAL_CAPS_SENTENCE

TextField Example

```
TextField tf =
        new TextField("Enter Email: ",
            "",
            50,
            TextField. EMAILADDR |
                TextField. NON_PREDICTIVE)
```

DateField

- Allows users to enter dates
  - Implementation dependent

- Different modes
  - DATE displays an editable date.
  - TIME displays an editable time.
  - DATE_TIME displays both a date and a time.

DateField

```
public DateField(String label,
                 int mode)

public Date getDate()

public void setDate(Date date)
```

ChoiceGroup

- User selects item(s) from choices

- Very similar to Lists
  - Except no Implicit type
  - Instead use ChoiceGroup.POPUP

```
public ChoiceGroup(String label,
        int choiceType,
        String[] stringElements,
        Image[] imageElements)
```

Adding Commands To Items

- You can add commands to Items just like Displayables
  - The Command should be of type Command.ITEM
  - When an item is selected a command is shown

```
textField.addCommand(MY_CMD);
```

- However the listener interface is different:
  - The class of the object must implement the interface **`ItemCommandListener`**
    - **`ItemCommandListener`** defines **`commandAction()`**

Item Layout

- You can decide how an item is laid out on the form:
  - `item.setLayout(Item.LAYOUT_2 | … );`
- Alignment:
  - `Item.LAYOUT_LEFT`
  - `Item.LAYOUT_CENTER`
  - `Item.LAYOUT_RIGHT`
- Newline:
  - `Item.LAYOUT_NEWLINE_BEFORE`
  - `Item.LAYOUT_NEWLINE_AFTER`

Example Item Layout

```
Form form = new Form("Form Title");
StringItem stringItem = new
  StringItem("Label: ", "Value");
stringItem.setLayout(Item.LAYOUT_2 |
  Item.LAYOUT_CENTER |
  Item.LAYOUT_NEWLINE_AFTER);
form.append(stringItem);
```

(Other layout controls, see Apress.)

Creating a Form

```
public Form(String title)

public Form(String title,
            Item[] items)
```

Adding Items to a Form

- You add items sequentially to a Form
  - Items are added top to bottom, left to right
  - If more items that screen, scroll bars are added

```
public int append(Item item);
```

  - Add the item to the form
  - Return the index it was added at
  - `append(String)`: append just a label

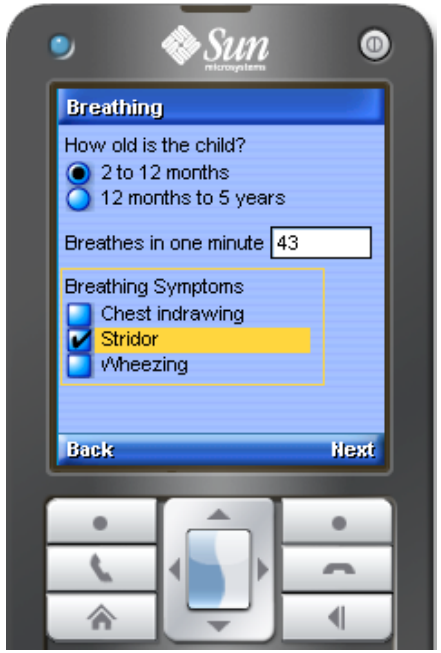
Example

```
look = new ChoiceGroup("How old is the child?",ChoiceGroup.EXCLUSIVE);
look.append("2 to 12 months",null);
look.append("12 months to 5 years",null);

TextField breaths =
   new TextField("Breaths in one minute","",5,TextField.NUMERIC);

ask = new ChoiceGroup("Breathing Symptoms",ChoiceGroup.MULTIPLE);
ask.append("Chest indrawing",null);
ask.append("Stridor",null);
ask.append("Wheezing",null);

form = new Form("Breathing");
form.append(look);
form.append(breaths);
form.append(ask);
form.addCommand(backCommand);
form.addCommand(nextCommand);
display.setCurrent(form);
```



More Form Fun

- `public void set(int index, Item item)`
- `public void insert(int index, Item item)`
- `public void delete(int index)`
- `public void deleteAll()`
- `public int size() public`
- `Item get(int index)`