Prior to subscribing for, installing or using any Third Party Products and Services it is your responsibility to ensure that your airtime service provider has agreed to support all of their features. Some airtime service providers may not offer Internet browsing functionality with a subscription to BlackBerry® Internet Service. Check with your service provider for availability, roaming arrangements, service plans and features. Installation or use of Third Party Products and Services with RIM's products and services may require one or more patent, trademark, copyright or other licenses in order to avoid infringement or violation of third party rights. You are solely responsible for determining whether to use, Third Party Products and Services and if any third party licenses are required to do so. If required you are responsible for acquiring them. You should not install or use Third Party Products and Services until all necessary licenses have been acquired. Any Third Party Products and Services that are provided with RIM's products and services are provided as a convenience to you and are provided "AS IS" with no express or implied conditions, endorsements, guarantees, representations or warranties of any kind by RIM and RIM assumes no liability whatsoever, in relation thereto. Your use of Third Party Products and Services shall be governed by and subject to you agreeing to the terms of separate licenses and other agreements applicable thereto with third parties, except to the extent expressly covered by a license or other agreement with RIM.

Certain features outlined in this documentation require a minimum version of BlackBerry® Enterprise Server, BlackBerry® Desktop Software, BlackBerry® Device Software and/or BlackBerry Handheld Software and may require additional development or Third Party Products and Services for access to corporate applications.

Certain products mentioned in this documentation include software developed by the Apache Software Foundation (http://www.apache.org/) and/or licensed pursuant to Apache License, Version 2.0 (http://www.apache.org/licenses/). For more information, see the NOTICE.txt file included with the software. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

The terms of use of any RIM product or service are set out in a separate license or other agreement with RIM applicable thereto. NOTHING IN THIS DOCUMENTATION IS INTENDED TO SUPERSEDE ANY EXPRESS WRITTEN AGREEMENTS OR WARRANTIES PROVIDED BY RIM FOR PORTIONS OF ANY RIM PRODUCT OR SERVICE OTHER THAN THIS DOCUMENTATION.
Chapter 6
User interface design for mobile devices

Objectives

- Explain why UI design is important when developing applications for mobile devices
- List the high-level best practices for designing a UI for mobile devices
- Explain the differences between MIDP and BlackBerry® UI classes.
- Describe the elements that you can use to design a UI for mobile devices
- Explain how to create UI elements to design a UI for mobile devices.

This chapter provides an overview of why UI design is important when developing media applications for mobile devices. This chapter outlines the limitations of mobile devices as compared to computers. This chapter outlines high-level best practices for developing the UI on a mobile device. This chapter also describes the following considerations when creating a UI for mobile devices: user interaction, screens, menus, managers, and fields.
The importance of UI design when developing applications for a mobile device

Thoughtful UI design can provide an effective BlackBerry smartphone user experience. This positive experience can contribute to user adoption of your application.

To design an effective UI, you must consider the needs of your audience and the nature of the mobile device. The challenge is to meet the action-oriented needs of the user within the limitations of the mobile device.

Applications designed for BlackBerry smartphones must provide a balance between the best possible user experience and a long battery life. When you design your BlackBerry smartphone application, consider the differences between mobile devices and computers. Mobile devices have the following limitations:

- have a smaller screen size that can display a limited number of characters
- display one screen at a time
- do not have a full keyboard
- have slower processor speeds
- use wireless networks, which have limited coverage and a longer latency period than standard LANs
- have less available memory
- have shorter battery life

Mobile device users use applications on their mobile device differently than they use applications on a computer. On mobile devices, users expect to find relevant information quickly and easily. For example, a user can check the local traffic conditions to find the quickest route to a destination. If the application takes a long time to load or display information, the user becomes frustrated and cannot accomplish the task.

Best practices for developing the UI for a mobile application

When you design applications for BlackBerry smartphones, be as consistent as possible with other BlackBerry smartphone applications. Consider the following guidelines:

- Use or extend existing UI components where possible so that your application can inherit the default behavior of the component.
- Follow the standard navigation model as closely as possible so that a particular user action produces a consistent result across applications.

When you design your application, also consider the following guidelines:
• Stay focused on the users’ immediate task. Simplify data selection and presentation by displaying only the information that users need at the moment.
• Ensure that you design the menu with actions that are relevant to users’ current context.
• Minimize the number of times that users need to click to complete a task.
• Design your UI to allow users to change their mind and undo commands. For example, always display a dialog box if users click a destructive menu item or button. Users sometimes click the wrong menu item or button accidentally.
• Display information in a way that makes effective use of the small screen.

Consistency of mobile device application UIs

To provide a positive user experience, ensure that the UI is consistent and predictable throughout your application. Also ensure that your application is consistent with how the BlackBerry smartphone already works.

Before you design your application, consider using the core applications on the BlackBerry smartphone or the BlackBerry Smartphone Simulator to learn more about the navigation model and best practices for designing the UI for your application.

When creating the UI of a BlackBerry Java Application, you can use the BlackBerry UI APIs. When designing MIDlets, use the MIDP UI APIs.

The BlackBerry UI APIs are a library of UI components that are designed to provide default layouts and behaviors that are consistent with BlackBerry Device Software applications.

• Screen components provide a standard screen layout, a default menu, and standard behavior after users press the Escape key, click the trackwheel or trackball, or click the screen.
• Layout managers provide an application with the ability to arrange components on a BlackBerry smartphone screen in standard ways, such as horizontally, vertically, or in a left-to-right flow.
• Field components provide standard UI elements for date selection, options, check boxes, lists, text fields, labels, and progress bar controls.

You can use the BlackBerry UI APIs to create UIs that include tables, grids, and other specialized features. The BlackBerry APIs use a standard Java event model to receive and respond to specific types of events. Applications can receive and respond to BlackBerry smartphone user events, such as after the BlackBerry smartphone user clicks the trackwheel, clicks the trackball, or types on the keyboard.

For more information about designing web pages for BlackBerry smartphones, see the BlackBerry Browser Content Design Guidelines.
APIs

You must select your UI APIs according to the type of application you want to create. If you design a UI for a BlackBerry Java Application, you use the BlackBerry UI APIs. If you design MIDlets, use the MIDP UI APIs.

A BlackBerry Application is an application that uses the BlackBerry APIs, as well as the CLDC APIs and possibly the MIDP APIs. A BlackBerry API application extends the `UiApplication` class and starts with a standard `main()` method.

Most of the BlackBerry® Java® Development Environment sample applications are BlackBerry API applications. All the BlackBerry Device Software applications (including the message list, contact list, calendar, and BlackBerry® Browser) are BlackBerry API applications.

Advantages of using the BlackBerry APIs include the following:

- BlackBerry UI APIs provide more functionality and flexibility than the standard `javax.microedition.lcdui` APIs that you use to create MIDlets.
- Applications can run background threads after the application closes.
- Applications can start automatically in the background when the device turns on.
- Applications can use IPC APIs to exchange information with other applications.
- Developers can create library modules that BlackBerry API applications can import.
- Applications can access trackball and trackwheel events.
- Applications can access **touch screen** events on a BlackBerry smartphone with a touch screen.
- Applications can use data from the accelerometer on a BlackBerry smartphone.
- Applications can integrate with BlackBerry Browser, BlackBerry® Maps, and other applications.
- Applications can change the icon that displays on the Home screen on a BlackBerry smartphone.

The only disadvantage of using BlackBerry APIs is that a BlackBerry API application can run only on a BlackBerry smartphone.
Considerations for UI design on mobile devices

Key considerations when designing a UI on mobile devices include the following:

• How will the BlackBerry smartphone user interact with the mobile device (for example, using touch screen, trackball, or keyboard)?
• What types of screens and screen elements will you use in the program?
• How will you arrange the elements on your screen?
• What type of controls will you use (for example, buttons or text fields)?
• How will you design menus?

User interaction

Depending on the model, users can interact with BlackBerry smartphones in the following ways:

• touch screen
• trackball or trackpad
• trackwheel
• keyboard

The type of selection method varies, depending on if the user has a touch screen, trackball, trackpad, or trackwheel on their BlackBerry smartphone. To provide an effective user experience, design your UI to ensure that it functions properly for different methods of user interaction.

Touch screen devices

You can optimize the user experience with the application on a BlackBerry smartphone with a touch screen by considering the following items:

• Users interact with the BlackBerry smartphone by touching the screen with their finger.
• Shortcut bars appear at the bottom of most screens.
• Users type text on a virtual keyboard.
• Users can view application screens in portrait mode or landscape mode.
• Some cursors, controls, and selection methods are different from cursors, controls, and selection methods on other BlackBerry smartphones.
• The screen, banner, background images, and icons on a BlackBerry smartphone with a touch screen are larger than other BlackBerry smartphones.

This screen shows a BlackBerry smartphone touch screen.
On BlackBerry smartphones with a trackball or trackpad, the trackball or trackpad is the primary control for user navigation. The trackball can move left, right, up, or down. Users can perform the following actions:

- Roll the trackball or slide a finger on the trackpad to move the cursor.
- Click the trackball or trackpad to perform default actions or open a context menu.
- Click the trackball or trackpad while pressing the Shift key to select text or select messages in a message list.

BlackBerry smartphones with a trackball or trackpad also include a Menu key that is located to the left of the trackball or trackpad. Users can press the Menu key to open a full menu of available actions.

This illustration shows the keys and features of a BlackBerry smartphone with SureType® technology and trackball.
BlackBerry smartphones that precede the BlackBerry® Pearl™ 8100 Series use a trackwheel as the primary control for user navigation. The trackwheel is located on the right side of the BlackBerry smartphone. The trackwheel moves up or down only.

Users can perform the following actions:

- roll the trackwheel to move the cursor vertically
- roll the trackwheel while pressing the Alt key to move the cursor horizontally
- click the trackwheel to select objects or open the menu
This illustration shows the keys and features of a BlackBerry smartphone with full QWERTY keyboard and trackwheel.

![BlackBerry smartphone with full QWERTY keyboard and trackwheel](image)

**Figure 7.3** BlackBerry smartphone with full QWERTY keyboard and trackwheel

**Keyboard**

Users use the keyboard primarily to type text. On BlackBerry smartphones without a touch screen, users can also use the keyboard to move around a screen (for example, to move around a map). However, navigation using the keyboard is always an alternative to navigation using the trackwheel, trackball, or trackpad.

BlackBerry smartphones have either a QWERTY keyboard or SureType keyboard. Both types of keyboard include character keys and modifier keys. Character keys send a character to the BlackBerry smartphone and include text keys, the Menu key, and the Escape key. A modifier key alters the functionality of character keys. Modifier keys include the Shift key and the Alt key.

On BlackBerry smartphones with a SurePress™ touch screen, in most cases, the QWERTY keyboard appears in landscape view and the SureType keyboard appears in portrait view.
**QWERTY keyboard**

The QWERTY keyboard provides users with a keyboard that is similar to the keyboard on a computer. Users type as they do on a computer keyboard except that numbers and symbols are in different places.

To type common symbols, users press the Alt key and the appropriate character key. To type other symbols, users press the Symbol key. After users press a modifier key, a typing mode indicator appears in the upper-right corner of the screen.

**SureType keyboard**

BlackBerry smartphones with SureType technology integrate a phone keypad and a familiar QWERTY-style keyboard with intuitive software. SureType technology is designed to predict words as users type them.

Users who use a BlackBerry smartphone with SureType technology can type text using the SureType input method or the multi-tap input method.

If users use SureType technology, they press the letter key for each letter in a word once. For example, to type *run*, users press the ER key once, the UI key once, and the BN key once. As users type, a list of possible letter combinations and words appears on the screen. SureType technology selects letter combinations or words from the list based on context. If users type a word or letter combination (for example, a web address or an abbreviation) that SureType technology does not recognize, users use the list that appears on the screen to build the word letter by letter. SureType technology is designed to learn new words or letter combinations that users type and add them to a custom word list.

If users use the multi-tap input method, they press a letter key once to type the first letter on the key and twice to type the second letter. For example, to type *run*, users press the ER key twice, the UI key once, and the BN key twice.

SureType technology is the default input method for most fields. Multi-tap is the default input method for phone number fields and password fields. If users switch from using SureType technology to using the multi-tap input method, a typing indicator appears in the upper-right corner of the screen to indicate the current input method.

**Shortcut keys**

On BlackBerry smartphones with a trackwheel, trackball, or trackpad, shortcut keys provide users with direct access to common actions within an application. Users can use shortcut keys for common actions such as composing an email message or adding a browser bookmark. Users can also use shortcuts to access buttons in a dialog box.

Application menu items do not have shortcut keys assigned to them. After users open the menu, they can press the first letter of a menu item to highlight the menu item, and then press the Enter key or click the trackwheel, trackball, or trackpad to select the menu item. If multiple menu items have the same first letter, users can continue pressing the letter until they highlight the menu item that they want.
Quiz

1. Which of the following UI elements controls the way elements are arranged on a BlackBerry smartphone?
   A. Screen component
   B. Field component
   C. Layout manager

2. Which of the following APIs can you use in a BlackBerry API application?
   A. net.rim.device.api.ui.Screen
   B. net.rim.device.api.ui.Manager
   C. net.rim.device.api.ui.component.ButtonField
   D. All of the above

3. Which of the following items apply to BlackBerry smartphones with a touch screen? Choose two.
   A. Users can view application screens in portrait mode only.
   B. Users can view application screens in portrait mode or landscape mode.
   C. Users can use a trackwheel as an alternative input method.
   D. Users type text on a virtual keyboard.

4. A QWERTY keyboard is designed to predict words as users type them. True or false?
   A. True
   B. False

5. Which of the following actions can you perform using shortcut keys? Choose two.
   A. Compose an email message
   B. Add a browser bookmark
   C. Access application menu items
Answers

1. C
2. D
A. B and D
3. B
4. A and B
GUI Hierarchy

The BlackBerry GUI in its most basic form is a three-level hierarchical structure, as follows.

The Screen class (net.rim.device.api.ui.Screen) is the starting point for the BlackBerry GUI. The BlackBerry smartphone displays only one screen at a time. The BlackBerry smartphone displays screens by pushing and popping them off of the display stack. The BlackBerry smartphone shows the user the screen on the top of the display stack. A screen can only exist once in the display stack, but the BlackBerry smartphone can push or pop it from the display stack at any time. The net.rim.device.api.ui.container package contains common sub classes of the Screen class.

The Manager class (net.rim.device.api.ui.Manager) manages the layout and interaction between field objects you place in it. With a manager you can specify the location and layout of fields it contains, scrolling, and focus change between fields. Every screen must contain at least one manager. The net.rim.device.api.ui.container package contains common sub classes of the Manager class.

The Field class provides the fundamental functionality for all field components. A field is a rectangular region that displays output to a user. A field can also handle input, and you must choose if you want to show the input to a user. The net.rim.device.api.ui.component package contains a library of prebuilt GUI components and controls that you can use to construct BlackBerry applications.

This illustration shows the hierarchy of BlackBerry GUI classes in which Screen classes contain Manager classes, and Manager classes contain Field classes.

Figure 7.4 GUI hierarchy
Screens

The main structure for a BlackBerry smartphone UI is the `Screen` object. A BlackBerry Java Application can display more than one screen at a time, but only one screen in a BlackBerry Java Application is active at one time.

The UI APIs initialize simple `Screen` objects. After you create a screen, you can add fields and a menu to the screen and display it to the BlackBerry smartphone user by pushing it on to the UI stack. The `Menu` object has associated menu items that are runnable objects, which perform a specific task when the BlackBerry smartphone user selects one of the items. For example, menu items can invoke the necessary code to establish a network connection, commit a data object to memory, or close the BlackBerry Java Application. For more sophisticated custom BlackBerry Java Applications, you can customize the BlackBerry smartphone UI and implement field types, as required. You can also add custom navigation and trackwheel behavior.

The `Screen` class does not implement disambiguation, which is required for complex input methods, such as international keyboards and the BlackBerry® 7100 Series. For seamless integration of the different input methods, extend Field or one of its subclasses. Do not use `Screen` objects for typing text.

To create a screen, do the following:

1. Import the following classes:
   ```java
   net.rim.device.api.ui.Screen
   net.rim.device.api.ui.container.FullScreen
   net.rim.device.api.ui.container.MainScreen
   ```

2. Extend the `Screen` class or one of its subclasses, `FullScreen` or `MainScreen`.

<table>
<thead>
<tr>
<th>Screen type</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Screen</td>
<td>Use the <code>Screen</code> class to use a manager to lay out UI components on the screen and to define a specific type of screen using the styles that the constants on the <code>Field</code> superclass define.</td>
</tr>
<tr>
<td>Standard vertical</td>
<td><code>FullScreen</code></td>
<td>By default, a <code>FullScreen</code> class contains a single vertical field manager. Use a <code>FullScreen</code> class to provide an empty screen that you can add UI components to in a standard vertical layout. For another layout style, such as horizontal or diagonal, use a <code>Screen</code> class and add a manager to it.</td>
</tr>
</tbody>
</table>
When designing the UI for your applications, you must consider both the screen size and screen display methods of BlackBerry smartphones.

### Screen size

BlackBerry smartphones have a smaller screen size than computer monitors. Because of this smaller screen size, BlackBerry smartphones can display a limited number of characters. BlackBerry smartphone screens also have less space for icons and other screen components.

Depending on the model, BlackBerry smartphone screens vary in size and layout (portrait and landscape). You must test your application UI to ensure that it displays properly on different BlackBerry smartphone models. For example, the display screen size on a BlackBerry Pearl 8100 Series smartphone is 240 by 260 pixels, while the display screen size on a BlackBerry Curve™ 8900 smartphone is 480 by 360 pixels. Also, on BlackBerry smartphones with a touch screen, users can switch between portrait view and landscape view by turning the device sideways.

### Screen display

BlackBerry smartphones display one screen at a time because they have a smaller screen size than computer monitors.

Common UI elements that you can display on the BlackBerry smartphone include the following:

- application screens
- banners
• dialog boxes
• progress indicators

Application screens

A BlackBerry smartphone can have multiple screens open at the same time, but users can only interact with one screen in a BlackBerry Java Application at one time. If users open multiple screens, the BlackBerry smartphone organizes the screens in a stack. The screen at the top of the stack is the active screen. If an application displays a screen, the BlackBerry Java Virtual Machine pushes it to the top of the stack. When an application closes a screen, the BlackBerry Java Virtual Machine deletes the screen from the top of the stack and displays the next screen on the stack, redrawing it as necessary.

An application screen can include the following elements: nonscrolling title bar, scroll arrows or bars, context menu, full menu, and shortcut bar.

<table>
<thead>
<tr>
<th>Screen element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonscrolling title bar</td>
<td>This bar appears at the top of the screen and can include a descriptive title of the screen.</td>
</tr>
<tr>
<td>scroll arrows or bars</td>
<td>If content extends beyond the viewing area, scroll arrows or scroll bars appear. Vertical scroll arrows or scroll bars appear on the right side of the screen, and horizontal scroll bars appear at the bottom of the screen. Scroll bars do not interfere with text.</td>
</tr>
<tr>
<td>context menu</td>
<td>A context menu contains a list of the most common available actions that users can perform within the current context.</td>
</tr>
<tr>
<td></td>
<td>On BlackBerry smartphones with a trackball or trackpad, a context menu appears if users click the trackball or trackpad and there is no default action. The context menu appears at the bottom of a screen, centered horizontally.</td>
</tr>
<tr>
<td></td>
<td>On BlackBerry smartphones with a touch screen, a context menu appears if users click an item that has a number of common actions associated with it (for example, a date separator in a message list or a contact name in a message). The context menu appears to the right of the highlighted item. If there is not enough space to the right, the context menu appears to the left of the highlighted item.</td>
</tr>
<tr>
<td>full menu</td>
<td>A full menu contains all the available actions for the application. To open a full menu, users press the Menu key. Full menus appear in the lower-left corner of the screen.</td>
</tr>
</tbody>
</table>
This screen shows features of a BlackBerry smartphone application screen.

| shortcut bar | On BlackBerry smartphones with a touch screen, a shortcut bar appears at the bottom of the screen. |

This screen shows features of a BlackBerry smartphone application screen.

Figure 7.5 Application screen

Dialog boxes

Use dialog boxes to perform the following actions:

- Prompt users for information that the application requires to complete a user-initiated task.
- Inform users of urgent information or the status of important actions.
- Warn users of unexpected or potentially destructive conditions or situations.

A dialog box includes an indicator that indicates the type of dialog box, a message, and buttons that allow users to perform an action. The size of the dialog box depends on the size of the BlackBerry smartphone screen. Scroll arrows appear if necessary. The theme that users select on their BlackBerry smartphone determines the visual style of the dialog box.

This screen shows a dialog box.

Figure 7.6 Dialog box
Different types of dialog boxes include OK, Delete, Save, Cancel, and Yes/No.

To create a dialog box, do the following:

1. **Import the** \texttt{net.rim.device.api.ui.component.Dialog} \texttt{class}.
2. **Create an alert dialog box specifying the alert text that you want to display.**
   
   \begin{verbatim}
   Dialog.alert("Specify the alert text that you want to display.
   \end{verbatim}

### Progress indicators

Use progress indicators to indicate the status of an operation. The two types of progress indicators include definite and indefinite progress indicators.

- Use **definite progress indicators** if you can determine the duration of an operation. Definite progress indicators include a label to indicate what the operation is and a horizontal bar that fills from left to right as an operation progresses. A percentage appears in the bar to indicate how much of the operation is complete. In the browser, progress indicators also indicate the number of kilobytes out of a total number of kilobytes that a BlackBerry smartphone downloads. To hide a definite progress indicator, users press the End key. If users press the End key, the operation continues, but users can perform other tasks at the same time.

- Use an **indefinite progress indicator** if you cannot determine the duration of an operation. You can indicate progress using a status dialog box. Users cannot perform other actions within the application while an application displays an indefinite progress indicator.

These screens show the differences between definite and indefinite progress indicators.

\begin{itemize}
  \item \textbf{Figure 7.7} Image of definite progress indicator
  \item \textbf{Figure 7.8} Image of indefinite progress indicator
\end{itemize}

To create a progress indicator, do the following:

1. **Import the** \texttt{net.rim.device.api.ui.component.GaugeField} \texttt{class}.
2. **Create an instance of a GaugeField.**
GaugeField percentGauge = new GaugeField("Percent: ", 1, 100, 29, GaugeField.PERCENT);

**Menus**

The Application Menu Item API, in the net.rim.blackberry.api.menuitem package, allows you to add menu items to BlackBerry Java Applications. The ApplicationMenuItemRepository class allows you to add or remove menu items from BlackBerry Java Applications.

With this code, you can create menus that users can use to perform many of the actions in an application. The full menu includes all the actions that users can perform in the application. After users press the Menu key, the full menu appears in the lower-left corner of the screen. Scroll arrows appear if the application offers more menu items that the user cannot see on the screen.

This screen shows a menu.

![Menu Screen](image)

**Figure 7.9 Menu**

To create a menu, do the following:

1. Import the required classes and interfaces.
   
   `import net.rim.device.api.ui.*;`
   
   `import net.rim.device.api.ui.component.*;`
   
   `import net.rim.device.api.ui.container.*;`

2. Create the application framework by extending the `UiApplication` class. In `main()`, create an instance of the new class and invoke `enterEventDispatcher()`, to enable the application to receive events. In the constructor, invoke `pushScreen`, to display the custom screen for the application. The `CreateMenuScreen` class represents the custom screen.

   ```java
   public class CreateMenu extends UiApplication {
   
   public static void main(String[] args)
   ```
3. **Create the custom screen for the application by extending the `MainScreen` class.** In the constructor, invoke `setTitle()`, to display the title for the screen. Invoke `add()`, to display a text field on the screen. Invoke `addMenuItem()`, to add a menu item to the default menu that is included with `MainScreen`.

```java
class CreateMenuScreen extends MainScreen
{
    public CreateMenuScreen()
    {
        setTitle("Create Menu Sample");
        add(new RichTextField("Create a menu"));
        addMenuItem(_viewItem);
    }
}
```

4. **Create the menu item by using the `MenuItem` class.** Override `run()`, to specify the action that occurs when the user clicks the menu item. When the user clicks the menu item, the application invokes `Menu.run()`.

```java
private MenuItem _viewItem = new MenuItem("More Info", 110, 10)
{
    public void run()
```
{ 
    Dialog.inform("Display more information");
}

};

5. Override close(), to display a dialog box to the user when the user clicks the Close menu item. The Close menu item is included by default. Invoke super.close(), to close the application. When the user closes the dialog box, the application invokes MainScreen.close(), to close the application.

public void close()
{
    Dialog.alert("Goodbye!");
    super.close();
}

Context menus

With this code, you can create a context menu, or short menu, which contains a list of the most common available actions that users can perform within the current context.

This screen shows a context menu.

Figure 7.10 Context menu

BlackBerry smartphones that have a trackball and menu key use two menu categories: primary action menus and full menus. Users can click a trackball to perform one of two actions: execute the default action (such as opening an email message), or display the primary actions menu (short menu). Users can press the menu key to display the full menu (default menu), which contains all menu items for the current screen. To build a primary actions menu, you must place a check in the makeMenu() method so the application can determine which menu to invoke.
Quiz

1. Which of the following objects is the main structure for a BlackBerry smartphone UI?
   A. Manager
   B. Screen
   C. Menu

2. Which of the following classes provides features that are common to standard BlackBerry Java Applications?
   A. Screen
   B. FullScreen
   C. MainScreen

3. Which of the following statements about the BlackBerry smartphone with a touch screen is false?
   A. Has limited space for text, icons, and other screen components
   B. Can display multiple screens at the same time
   C. Can display in both portrait or landscape mode

4. Which of the following items appears only on BlackBerry smartphones with a touch screen?
   A. Context menu
   B. Full menu
   C. Shortcut bar
   D. Scroll arrows

5. Which actions does a context menu display in an application?
   A. All the actions
   B. The primary actions
   C. The most commonly used actions
   D. The most recently used actions
Answers

1. B
2. C
3. B
4. C
5. C
Managers

The Manager class is the next level in the BlackBerry GUI hierarchy. Managers build on top of screens. You use managers for scrolling (vertical and horizontal) and to position and lay out fields. You add fields to managers to place them in an appropriate area on the screen. The BlackBerry API set contains a set of managers that extend from the Manager class and provide a layout mechanism for common screen designs. Managers include the following:

- **HorizontalFieldManager**: A manager that lays out fields from left to right in a single row. This manager can provide horizontal scrolling for fields that do not fit on the screen as well as vertical scrolling for fields that are taller than the screen.

- **VerticalFieldManager**: A manager that lays out fields in a single vertical row. This manager can provide vertical scrolling for fields that do not fit on the screen as well as horizontal scrolling for fields that are wider than screen.

- **FlowFieldManager**: A manager that lays out fields in a horizontal then vertical flow. Fields are positioned from left to right. This manager places any fields that do not fit in the allotted horizontal space on the next line below, beginning from the left. The FlowFieldManager also supports both horizontal and vertical scrolling.

- **DialogFieldManager**: This manager handles an icon, a message, and a special area, which can hold a list of user-specified custom fields. Use a VerticalFieldManager to lay out the fields in the user area. A VerticalFieldManager lays out its icon in the top left corner, and its message label in the top left corner.

To create a layout manager, do the following:

1. Import the following classes:
   - net.rim.device.api.ui.Manager
   - net.rim.device.api.ui.container.HorizontalFieldManager
   - net.rim.device.api.ui.component.ButtonField

2. Create an instance of a HorizontalFieldManager and HorizontalFieldManager.
   ```java
   _fieldManagerBottom = new HorizontalFieldManager();
   ```

3. Invoke the add() method to add the HorizontalFieldManager to a screen.
   ```java
   myScreen.add(_fieldManagerBottom);
   ```

4. Create an instance of a ButtonField.
   ```java
   ButtonField mySubmitButton = new ButtonField("Submit");
   ```

5. Add the ButtonField to the HorizontalFieldManager.
   ```java
   _fieldManagerBottom.add(mySubmitButton);
   ```
Nesting managers

Managers are an extension of the Field class. This means managers can also contain other managers. This is known as nesting. You can use nesting to create various enhanced layout styles on the BlackBerry smartphone; such as a column or table layout.

If you nest one manager into another manager, you can create various field layouts on the screen. If you select managers that enable scrolling, you can also control how a user interacts with the fields on the screen.

This screen shows a simple example of nesting managers.

<table>
<thead>
<tr>
<th>User List</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
</tr>
<tr>
<td>Homer</td>
</tr>
<tr>
<td>Marge</td>
</tr>
<tr>
<td>Bart</td>
</tr>
<tr>
<td>Lisa</td>
</tr>
<tr>
<td>Maggie</td>
</tr>
</tbody>
</table>

*Figure 7.11 Nesting managers*

In this nesting managers example, the developer created two columns with two vertical field managers that contain label fields. A horizontal field manager contains these two columns. A larger vertical field manager contains the entire layout.

This illustration shows the underlying structure of the managers and fields used in the previous example.
Figure 7.12 Structure of nesting managers
Fields

Field components provide standard UI elements for controls such as date selection, options, check boxes, lists, text fields and labels, and progress bars.

Fields represent all UI components, which are rectangular regions that a Manager contains. The layout requirements for a field determine the size of the field. Managers provide scrolling for the fields that they contain.

To create a specialized field component (for example, a text field that contains multiple elements), create your own custom types by extending the Field class or one of its subclasses.

<table>
<thead>
<tr>
<th>Traditional field</th>
<th>BlackBerry field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>ButtonField</td>
</tr>
<tr>
<td>Check box</td>
<td>CheckboxField</td>
</tr>
<tr>
<td>Date</td>
<td>DateField</td>
</tr>
<tr>
<td>Dialog box</td>
<td>PopupScreen</td>
</tr>
<tr>
<td>Drop-down list</td>
<td>NumericChoiceField or ObjectChoiceField</td>
</tr>
<tr>
<td>Radio button</td>
<td>RadioButtonField</td>
</tr>
<tr>
<td>Text</td>
<td>RichTextField, BasicEditField, EditField, PasswordEditField, or AutoTextEditField</td>
</tr>
<tr>
<td>Text label</td>
<td>LabelField</td>
</tr>
<tr>
<td>List</td>
<td>ListField</td>
</tr>
</tbody>
</table>

When designing controls for the BlackBerry smartphone UI, you must select an appropriate control and ensure that the control works with different user interaction methods.

Buttons

When you add this code, an action occurs after users click a button.

This screen shows a button.
User interface design for mobile devices

Figure 7.13 Buttons

To create a button, do the following:

1. Import the `net.rim.device.api.ui.component.ButtonField` class.

2. Create an instance of a `ButtonField` using a style parameter.

   ```java
   ButtonField mySubmitButton = new ButtonField("Submit");
   ```

Check boxes

With this code, you can create check boxes for options that users can turn on or turn off.

This screen shows check boxes.

Figure 7.14 Check boxes

To create check boxes, do the following:

1. Import the `net.rim.device.api.ui.component.CheckboxField` class.

2. Create an instance of a `CheckboxField`.

   ```java
   CheckboxField myCheckbox = new CheckboxField
   ```
Drop-down lists

With this code, you can create drop-down lists to provide a set of mutually exclusive values.

This screen shows a drop-down list.

To create drop-down lists, do the following:

1. Import the following classes:
   - java.lang.String
   - net.rim.device.api.ui.component.ObjectChoiceField

2. Create an instance of an ObjectChoiceField, providing an object array as a parameter to create a drop-down list that contains objects.

   String choiceItems[] = {"Option one","Option two","Option three"};

   ObjectChoiceField myObjectChoiceField = new ObjectChoiceField("Pick One: ", choiceItems);

List boxes

With this code, you can create a list box to display a list from which users can select one or more values.

This screen shows a list box.
**Figure 7.16 List box**

To create list boxes, do the following:

1. Import the following classes:
   - `java.lang.String`
   - `net.rim.device.api.ui.component.ListField`
   - `net.rim.device.api.ui.container.MainScreen`

2. Import the `net.rim.device.api.ui.component.ListFieldCallback` interface.

3. Create a class that implements the `ListFieldCallback` interface.
   ```java
   private class ListCallback implements ListFieldCallback {
   ```

4. Create the items that you want to display in a `ListField`.
   ```java
   String fieldOne = new String("Marco Cacciacarro");
   String fieldTwo = new String("Meredith Wagler");
   ```

5. Create an instance of a `ListField`.
   ```java
   ListField myList = new ListField(0, ListField.MULTI_SELECT);
   ```

   ```java
   ListCallback myCallback = new ListCallback();
   ```

7. Set the call back of the `ListField` to be the `ListCallback`.
   ```java
   myList.setCallback(myCallback);
   ```

8. Use the `ListCallBack` object to add items to the `ListField`.
   ```java
   myCallback.add(myList, fieldOne);
   myCallback.add(myList, fieldTwo);
   ```

9. Add the `ListField` to the `MainScreen`.
   ```java
   mainScreen.add(myList);
   ```

**Option buttons**

With this code, you can create option buttons to indicate a set of mutually exclusive but related choices.
This screen shows option buttons.

![Option buttons](image)

*Figure 7.17 Option buttons*

To create option buttons, do the following:

1. **Import the following classes:**
   - net.rim.device.api.ui.component.RadioButtonGroup
   - net.rim.device.api.ui.component.RadioButtonField

2. **Create an instance of a RadioButtonGroup.**
   ```java
   RadioButtonGroup rbGroup = new RadioButtonGroup();
   ```

3. **Create an instance of a RadioButtonField for each option you want to make available to the BlackBerry smartphone user.**
   ```java
   RadioButtonField rbField = new RadioButtonField("First option");
   RadioButtonField rbField2 = new RadioButtonField("Second option");
   ```

4. **Invoke RadioButtonGroup.add() to add the RadioButtonField instances to the RadioButtonGroup.**
   ```java
   rbGroup.add(rbField);
   rbGroup.add(rbField2);
   ```

**Text fields**

With this code, you can create a text field in which users can type text.

**Create a read-only text field that allows formatting**

To create a read-only text field that allows formatting, do the following:

1. **Import the net.rim.device.api.ui.component.RichTextField class.**
2. Create an instance of a RichTextField.

   RichTextField rich = new RichTextField("The text I want to show.");

Create an editable text field that has no formatting and accepts filters

To create an editable text field that has no formatting and accepts filters, do the following:

1. Import the following class:
   • net.rim.device.api.ui.component.BasicEditField

2. Create an instance of a BasicEditField.

   BasicEditField bf = new BasicEditField("BasicEditField: ", ",
   10, EditField.FILTER_UPPERCASE);

Create an editable text field that allows special characters

To create an editable text field that allows special characters, do the following:

1. Import the net.rim.device.api.ui.component.EditField class.

2. Create an instance of an EditField.

   EditField edit = new EditField("EditField: ", ",
   10, EditField.FILTER_DEFAULT);

Create a text field for AutoText

If a text field supports AutoText, when users press the Space key twice, the BlackBerry smartphone inserts a period, capitalizes the next letter after a period, and replaces words as defined in the AutoText application.

To create a text field for AutoText, do the following:

1. Import the following classes:
   • net.rim.device.api.ui.component.AutoTextEditField
   • net.rim.device.api.ui.autotext.AutoText
   • net.rim.device.api.ui.component.BasicEditField

2. Create an instance of an AutoTextEditField.

   AutoTextEditField autoT = new AutoTextEditField("AutoTextEditField: ", ");
Search fields

With this code, you can create search fields to allow users to search for items in an application. Several BlackBerry smartphone applications include search fields. For example, in the task list, users can use a search field to search for specific tasks.

As users type text in a search field, the application searches for and displays items that begin with the search text. If users type multiple words in the search field, the application displays results that match both words. For example, if users type “ca ba”, the application returns all items with a word that begins with “ca” and a word that begins with “ba” (such as “call back”).

This screen shows a search field.

![Figure 7.18 Search field](image)

You can create an application that uses the `KeywordFilterField` class, included in the `net.rim.device.api.ui.component` package, to provide a UI field that consists of a single text input field and a list of selectable elements. As users type text in a search field, the application filters the elements in the list that begin with the search text.

To create search fields, do the following:

1. Import the following classes:
   - `net.rim.device.api.ui.component.KeywordFilterField`
   - `net.rim.device.api.collection.util.SortedReadableList`
   - `java.util.Vector`
   - `java.io.InputStream`
   - `net.rim.device.api.io.LineReader`
   - `java.lang.String`

2. Import the `net.rim.device.api.ui.component.KeywordProvider` interface.

3. Create variables. In the following code sample, `CountryList` extends the `SortedReadableList` class and implements the `KeywordProvider` interface.

   ```java
   private KeywordFilterField _keywordField;
   private CountryList _CountryList;
   ```
4. To create a list of selectable text items, populate a vector with data from a text file.
   
   ```java
   _countries = getDataFromFile();
   ```

5. Create an instance of a class that extends the `SortedReadableList` class.
   
   ```java
   _CountryList = new CountryList(StringComparator.getInstance(true), _countries);
   ```

6. To specify the elements of the list, create a new instance of a `KeywordFilterField` object.
   
   ```java
   _keywordField = new KeywordFilterField();
   ```

7. Invoke `KeywordFilterField.setList()`.
   
   ```java
   _keywordField.setList(_CountryList, _CountryList);
   ```

8. Set a label for the input field of the `KeywordFilterField`.
   
   ```java
   _keywordField.setLabel("Search: ");
   ```

9. Create the main screen of the application and add a `KeywordFilterField` to the main screen.

   ```java
   KeywordFilterDemoScreen screen = new KeywordFilterDemoScreen(this, _keywordField);
   screen.add(_keywordField.getKeywordField());
   screen.add(_keywordField);
   pushScreen(screen);
   ```

10. To create a method that populates and returns a vector of Country objects containing data from text file, in the method signature, specify `Vector` as the return type.

    ```java
    public Vector getDataFromFile()
    ```

11. Create and store a reference to a new `Vector` object.

    ```java
    Vector countries = new Vector();
    ```

12. Create an input stream to the text file.

    ```java
    InputStream stream = getClass().getResourceAsStream("/Data/CountryData.txt");
    ```

13. Read CRLF delimited lines from the input stream.
LineReader lineReader = new LineReader

14. Read data from the input stream one line at a time until you reach the end of file flag. Each line is parsed to extract data that is used to construct Country objects.

for(;;){
    // Obtain a line of text from the text file
    String line = new String(lineReader.readLine());

    // If we are not at the end of the file, parse the line of text
    if(!line.equals("EOF")) {
        int space1 = line.indexOf(" ");
        String country = line.substring(0,space1);
        int space2 = line.indexOf(" ",space1+1);
        String population = line.substring(space1+1,space2);
        String capital = line.substring(space2+1,line.length());

        // Create a new Country object
        countries.addElement(new Country(country,population,capital));
    }
    else {
        break;
    }
}  // end the for loop

return countries;

To add a keyword to

15. To add a keyword to the list of selectable text items, invoke SortedReadableList.doAdd(element).
SortedReadableList.doAdd(((Country)countries.elementAt(i)).getCountryName()) ;

16. To update the list of selectable text items, invoke KeywordFilterField.updateList().
   _keywordField.updateList();

17. To obtain the key word that a BlackBerry smartphone user typed into the KeywordFilterField, invoke KeywordFilterField.getKeyword().
   String userTypedWord = _keywordField.getKeyword

**Tree views**

With this code, you can create a tree view to display objects, such as a folder structure, in a hierarchical manner.

Objects in the tree view are nodes. The highest node is the root node. A node in the tree can have child nodes under it. A node that has a child is a parent node.

This screen shows a tree view.

![Tree view](image)

*Figure 7.19 Tree view*

To create a field to display a tree view, do the following:

1. Import the following classes:
   - net.rim.device.api.ui.component.TreeField
   - java.lang.String
   - net.rim.device.api.ui.container.MainScreen

2. Import the net.rim.device.api.ui.component.TreeFieldCallback interface.

3. Implement the TreeFieldCallback interface.

4. Invoke TreeField.setExpanded() on the TreeField object to specify whether a folder is collapsible. Create a TreeField object and multiple child nodes to the TreeField object. Invoke TreeField.setExpanded() using node4 as a parameter to collapse the folder.
String fieldOne = new String("Main folder");
...
TreeCallback myCallback = new TreeCallback();
TreeField myTree = new TreeField(myCallback, Field.FOCUSABLE);
int node1 = myTree.addChildNode(0, fieldOne);
int node2 = myTree.addChildNode(0, fieldTwo);
int node3 = myTree.addChildNode(node2, fieldThree);
int node4 = myTree.addChildNode(node3, fieldFour);
...
int node10 = myTree.addChildNode(node1, fieldTen);
myTree.setExpanded(node4, false);
...
mainScreen.add(myTree);

5. To repaint a TreeField when a node changes, create a class that implements the TreeFieldCallback interface and implement the TreeFieldCallback.drawTreeItem method. The TreeFieldCallback.drawTreeItem method uses the cookie for a tree node to draw a String in the location of a node. The TreeFieldCallback.drawTreeItem method invokes Graphics.drawText() to draw the String.

private class TreeCallback implements TreeFieldCallback
{
    public void drawTreeItem(TreeField _tree, Graphics g, int node, int y, int width, int indent)
    {
        String text = (_tree).getCookie(node);
        g.drawText(text, indent, y);
    }
}
Quiz

1. Which of the following classes provides an application with the ability to arrange components on a BlackBerry smartphone screen?
   A. Screen
   B. Manager
   C. Field

2. Which control can you use to initiate an action?
   A. Check box
   B. Button
   C. Drop-down list

3. Which type of content do list boxes display?
   A. A set of mutually exclusive but related values
   B. A set of mutually exclusive values
   C. A set of values

4. Which type of control can you use to display objects in a hierarchical structure?
   A. Tree view
   B. List box
   C. Search field
   D. Any of the above
Chapter 6

Answers

1. B
2. B
3. C
4. A
Thoughtful UI design can provide an effective BlackBerry smartphone user experience. This positive experience can contribute to user adoption of your application. To design an effective UI, you must consider the action-oriented needs of the user and the limitations of the mobile device.

When you design applications for BlackBerry smartphones, be as consistent as possible with other BlackBerry smartphone applications. Optimize your application UI so that users can find relevant information quickly and easily.

The UI APIs you use depend on the type of application you want to create. When designing the UI of a BlackBerry Java Application, you can use the BlackBerry UI APIs. When designing MIDlets, use the MIDP UI APIs.

A BlackBerry Application is an application that uses the BlackBerry APIs, as well as the CLDC APIs and possibly the MIDP APIs. A BlackBerry API application extends the `UiApplication` class and starts with a standard `main()` method.

Key considerations when creating a UI for mobile devices: user interaction, screens, menus, managers, and fields.

To provide an effective user experience, design your UI to ensure that it functions properly for different methods of user interaction. The type of selection method varies, depending on if the user has a touch screen, trackball, trackpad, or trackwheel on their BlackBerry smartphone. Also, the input method for typing differs, depending on if the user has a QWERTY and SureType keyboard.

The BlackBerry GUI in its most basic form is a three-level hierarchical structure which uses the `Screen` class, `Manager` class, and `Field` class.

The `Screen` class (`net.rim.device.api.ui.Screen`) is the starting point for the BlackBerry GUI. The BlackBerry smartphone displays only one screen at a time. The BlackBerry smartphone displays screens by pushing and popping them off of the display stack.

The `Manager` class (`net.rim.device.api.ui.Manager`) manages the layout and interaction between field objects you place in it. With a manager, you can specify the location and layout of the fields it contains, scrolling, and focus change between fields. Every screen must contain at least one manager.

The `Field` class provides the fundamental functionality for all field components. A field is a rectangular region that displays output to a user. A field can also handle input, and you must choose if you want to show the input to a user. The `net.rim.device.api.ui.component` package contains a library of prebuilt GUI components and controls for constructing BlackBerry applications.
After you create a screen, you can add fields and a menu to the screen and display it to the BlackBerry smartphone user by pushing it on to the UI stack. You add these fields and menus to managers to place them in an appropriate area on the screen.

Users use menus to perform many of the actions in an application. The full menu includes all the actions that users can perform in the application. A context menu, or short menu, contains a list of the most common available actions that users can perform within the current context.

When designing controls for the UI, you must select an appropriate control and ensure that the control works with different user interaction methods. Field components provide standard UI elements for controls including buttons, check boxes, drop-down lists, list boxes, option buttons, text fields, search fields, and tree views.
Review Questions

1. List four key considerations for developing a UI for your mobile device applications.

2. List four different input devices a user can use to interact with a BlackBerry smartphone.

3. List the two types of keyboards used in BlackBerry smartphones and explain the differences between the ways these keyboards function.

4. Describe how BlackBerry smartphone screen sizes affect UI design.

5. Define the following objects:
   A. Screen
   B. Menu
   C. Field
   D. Manager

6. Explain when to use the following screen components:
   A. Dialog boxes
   B. Progress indicators

7. List eight types of controls and describe when to use them.

8. List two types of menus used in mobile devices and describe what you can use each type of menu for.