

**Exam Preparation Notes
Covering
MIDP and CLDC API
For
Sun Certified Mobile Application Developer
Sun Microsystems, Inc**

01 September, 2004

**Ahmed S. Habib Khan
(SCJP, SCWCD, OCP, MCP)
Ahmadkhan18@hotmail.com**

Description of J2ME classes in alphabetical order

Alert Class:

The image for Alert can be immutable or mutable

An alert can have a gauge as an indicator by calling `setIndicator (Gauge)`, the gauge must conform to all following restrictions:

- It must not be interactive
- It must not be owned by another container
- It must not have any commands
- It must not have any `ItemStateListener`
- It must not have a label that is the label must be null.
- Its preferred width and height must be unlocked
- The Layout value be `LAYOUT_DEFAULT`

There is always at least one command present on an alert `DISMISS_COMMAND`

If there are 2 or more commands on the alert, it is automatically turned modal and timeout value is always `FOREVER`.

If an alert has one command either `DISMISS_COMMAND` or another command, alert may have timed behavior.

At the time of creation an implicit `CommandListener` is associated with with alert which can be replace by user defined listener by calling `setCommandListener`

The `Displayable.setCurrent(Alert, Displayable)` is only effective if default listener is in place.

Canvas:

- Requires the application to subclass it as the `paint` method is abstract
- The event delivery methods are all called serially, that is the implementation will never call an event delivery method prior to any of the event delivery method returned. The service `Repaints` method is an exception of this rule, as it blocks until `paint()` is called and returns. This will happen even if the application is in the midst of a event delivery method when it calls the `serviceRepaints()`.
- The `Display.callSerially()` method can be used to serialize some application-defined work with the even stream.

- Key, pointer and paint method will only be called when the canvas is actually visible on the device output.
- The calls to showNotify and hideNotify are not under the control of the MIDlet
- The canvas is in normal mode by default, which can be change via setFullScreenMode(Boolean) method.
- The network usage indicator if rendered on the screen must be visible in full screen mode.
- The contents of canvas are never saved if it is hidden and then is made visible again. So after the showNotify the paint method will always be called.
- Application code must never call the paint() method, it is called by the implementation.
- Repaint will not wait for paint() to complete so it is asynchronous of repaint(). To synchronize use the Display.callSerially or repaint method.

Choice:

- The image part of the Choice element can be both mutable or immutable
- An application can query the recommended image size by the implementation using Display.getBestImageWidth() and Display.getBestImageHeight().
- IMPLICIT is only valid for List, while POP_UP is only valid for ChoiceGroup

CommConnection:

- The port identifier must not contain semi colon (;)
- If the requested baud rate is not supported on the platform, then the system may use an alternative valid setting.

Command:

- For Command types other than screen the implementation may replace the label with a system-specific label.
- If a long label is not present (which is optional), short label will be used.
- The implementation is allowed to switch between short and long label at will.

CommandListener:

- The actionPerformed() method should return immediately

Connector:

- READ_WRITE is the default connection mode
- The Connector open*Stream methods may throw the following exception:
 - **IllegalArgumentException**, if parameter is invalid

- **ConnectionNotFoundException**, if the requested protocol type is not supported or if the target name cannot be found
- **IOException**, if an I/O error occurs
- **SecurityException**, May be thrown if access to the protocol handler is prohibited.

ContentConnection:

- This interface defines the Stream connection over which content is passed.

Control:

- Control object provides a logical grouping of media processing functions.
- The same object can implement multiple controls.

CustomItem:

- Subclasses are responsible for their visual appearance, even handling and calling `Item.notifyStateChanged()`

Datagram:

- Since CLDC 1.0

DatagramConnection:

- A datagram connection can be opened in a "client" mode or "server" mode. If the host part is missing then the connection is opened as "server", when the host part is specified, the connection is opened as a "client".
- The port number in server mode is that of the receiving port. The port number in client mode is that of the target port. The reply-to-port is never unspecified In both cases. In server mode the same port number is used for both receiving and sending. In "client" mode the reply to port is always dynamically allocated.
- Datagram objects must be allocated using the `DatagramConnection.newDatagram()`, the resulting object is defined using another interface called `Datagram`

Date:

- `Date.toString` is new in CLDC 1.1

DateField:

- If the DateField.TIME is used the date part must be set to zero epoch and must not be accessed.
- In DateField.DATE mode the time component is set to zero
- If the TimeZone is null the system default TimeZone is used

Display:

- This is exactly one instance of a Display per MIDlet.
- Calling the Display.getDisplay() several time will return the same reference of the Display.
- One time operations should not occur in MIDlet.startApp, but in MIDlet's constructor.
- At any time the application may have at most one displayable that is intended to be shown on the device display, this Displayable is referred to as current Displayable.
- It is possible for getCurrent to return null, this may happen at the startup before the MIDlet application has called the setCurrent.
- The value returned by Displayable.isShown will return false if the current displayable is obscured by a system screen.
- The number of colors for a black and white display is 2
- callSerially() will never block
- flashBackLight and vibrate must return immediately and must not block, while the device is flashing or vibrating. Call to these methods will only be honored if the Displayable is in foreground, and will return false if the device does not support flashback light or vibration.

Displayable:

- The title string may contain line breaks.
- Initial state of a Displayable when created is
 - It is not visible on the Display
 - There is no Ticker associated with it.
 - The title is null
 - There are no commands present
 - There is no commandListener present

Form:

- An item may be placed at most within one form. If an application attempts to place an item into a form, which is already placed on another form, an IllegalStateException is thrown.
- The layout policy in a Form is organized around rows. All rows in a particular form have the same width.

- Rows need not to have all the same height
- If the layout direction is left-to-right then the initial alignment value must be LAYOUT_LEFT, else if the layout direction is right-to-left then the initial value must be LAYOUT_RIGHT.

GameCanvas:

- At creation time a new buffer is also created for GameCanvas and filled with white pixels.
- The Graphics object obtained via getGraphics renders only to off-screen buffer belonging to this canvas.
- Rendering operations do not appear on the display until the flushGraphics() is called. Flushing the buffer does not change or clear its contents.
- Each call to getGraphics returns a new Graphics object.
- For each GameCanvas all the Graphics objects will render to same off-screen buffer.
- A newly created Graphics object has the following properties:
 - The destination is this GameCanvas's buffer
 - The clip region encompasses the entire buffer.
 - The current color is black.
 - Font is the same as returned by Font.getDefaultFont()
 - The stroke style is SOLID
 - The origin of the coordinate system is located at the upper-left corner of the buffer.
- The getKeyStates() get the physical state of the game keys. A key bit will be one if the key is currently down or has been pressed at least since last time this method was called, other wise it will be zero. This method returns if the GameCanvas is not currently visible.
- The paint method will paint the off-screen buffer at (0,0).
- The flushGraphics() method does not return immediately until the flush has been completed. This method does nothing and returns if the GameCanvas is not visible or the system busy.

Gauge:

- Will throw an IllegalArgumentException if the maxvalue is not positive for interactive gauge, or if the maxvalue is not positive nor INDEFINATE for non-interactive Gauges, or if the gauge is not interactive and the maxvalue is INDEFINATE but the initial value is not one of the following: INCREMENTAL_IDLE, INCREMENTAL_UPDATING, CONTINUOUS_IDLE, CONTINUOUS_RUNNING

HttpConnection:

- The HttpConnection exists in one of the following states
 - Setup, in which the request parameter can be set
 - Connected, in which the request headers have been sent and an response is expected.
 - Closed, the final state in which the HttpConnection is closed.
- The following methods can only be called in setup mode:
 - SetRequestMethod
 - SetRequestProperty
- The transition from setup mode to connected mode is caused by any method that requires to send or receive data from the server.
- Once the openInputStream or openDataInputStream is called, calling the methods that can only be called in setup mode will be ignored. Once the request parameters are sent, these methods will throw IOException

Item:

- The default state of newly created Item is:
 - The item is not contained within any container
 - There are no commands present
 - The default command is null
 - The ItemStateListener is null
 - The default layout is LAYOUT_DEFAULT and both the preferred width and height are unlocked.

ItemStateListener:

- The itemStateChanged (Item) will be called when the user:
 - Changes the set of selected values in ChoiceGroup
 - Adjusts the values of interactive gauge
 - Enters or modifies the value of TextField (TextBox does not generate itemStateChanged events)
 - Enters a new date or time in DateField Or
 - Item.notifyStateChanged() is called on the Item.

LayerManager:

- A layers index correlates to its z-order, The layer at index 0 is closer to the user, the layer with the highest index is furthest away from user.
- The indices are always contiguous, if a layer is removed the indices of other layers will be adjusted to maintain the contingency.

MIDlet:

- A MIDlet can be in three states:
 - Active (entered when the startApp is called)
 - Paused (entered when the pauseApp is called, this method will only be called when the MIDlet is in active state)
 - Destroyed (entered when the destroyApp is called, this method can be called when the MIDlet is either in active or paused state).
- A MIDlet can request not to enter destroyed state by throwing MIDletStateChangeException, inside the destroyApp method. This is only valid if the unconditional flag is false. If it is true the MIDlet is assumed to be in destroy state regardless of how the destroyApp method terminates.
- notifyDestroyed method is called to inform the AMS that the MIDlet has entered the destroyed state. The AMS will not call the destroyApp method of MIDlet, and reclaim all resource allocated and used by this MIDlet.
- notifyPaused method can be called by a MIDlet to inform the AMS that it has entered the paused state. If the application pauses it self it must call resumeRequest to request to reenter the active state.
- platformRequest is a non-blocking method. In addition this method does not queue multiple requests. On platforms where the MIDlet suite must exit before the request is handled, the platform must handle only the last request made. On platforms where the MIDlet and the application can run together each request made must be passed to platform software for handling in a timely fashion.
- If the URL refers to a MIDlet suite (Either a Application descriptor or jar file), the application handling the request must interpret it as a request to install the named package and the user must be allowed to control the installation process like canceling, downloading and installing.
- If MIDlet suite being installed is an update of the currently running MIDlet, then the MIDlet must be stopped first before updating.

Player:

- **A player has five states and also moves in following order:**
 - 1. UNREALIZED**
 - A player starts in this mode, and does not have enough information to acquire all resources it needs. Following methods should not be called in this state as they will throw an `IllegalStateException`.
 - `getContentType`
 - `setMediaTime`
 - `getControls`
 - `getControl`
 - The `realize` method transitions the player from UNREALIZED state to REALIZED.
 - 2. REALIZED**
 - A Player is in REALIZED state when it has obtained the information required to acquire the media resources
 - After invoking `realize` on a player, the only way a Player can return back to UNREALIZED state is if `deallocate` is invoked before `realize` is complete. Once a player reaches the REALIZED state it never returns to UNREALIZED back.
 - 3. PREFETCHED**
 - Once realized a player still needs to perform time consuming tasks like startup operations, filling the buffers with media data. Calling `prefetch` carries out these tasks. Once a player is PREFETCHED state it may be started. When a player is stopped not closed it returns to PREFETCHED state. If a player is in PREFETCHED mode invoking `deallocate` will return the player back REALIZED state.
 - 4. STARTED**
 - A player can enter the STARTED state by calling `start`, once it is in PREFETCHED state. A player can be stop by invoking the `stop` method or if the end of media is reached, where it returns to PREFETCHED state. The `setLoopCount` method must not be called when the player is in STARTED state as an `IllegalStateException` will be thrown.
 - 5. CLOSED**
 - Calling the `close` method will put the player in CLOSED state. A player at any of the above mentioned states can call the `close` method. Calling `getControl` or `getControls` will throw an `IllegalStateException` in this state.

- To receive player events the `PlayerListener` must be implemented and register via `addPlayerListener()`.
- If `realize` is called when the player is in `REALIZED`, `PREFETCHED` or `STARTED` state it will be ignored.
- If `prefetch` is called when the player is in `UNREALIZED` state it will implicitly call `realize`. If `prefetch` is called when player is in `STARTED` mode, it will be ignored
- If a `Player` cannot acquire all the resources it needs, It will throw a `MediaException`, in this case the `Player` can not start, but the `prefetch` may be called later when the resources are available.
- If the `start` is called when the player is `UNREALIZED` or `REALIZED` state it will implicitly call `prefetch`. If `start` is called when the player is in `STARTED` mode it will be ignored.
- If `deallocate` is called when the player is IN `REALIZED` or `UNREALIZED` state it will be ignored.
- If `deallocate` is called when the `Player` is in `STARTED` mode, it will implicitly call stop on the player.
- The return value by `setMediaTime` is the accurate value set for media time, may not be the value passed. Setting media time to negative will set it to zero, otherwise if the media time is set beyond duration of the media, it will be set to end of media.
- For some media types the media time can not be set, It will throw a `MediaException`
- Passing zero as a loop count to `setLoopCount` will throw an `IllegalArgumentException`, setting it to `-1` will loop and play indefinitely
- `PlayerListener` has one method with following signature:

```
public void playerUpdate(Player player, String event, Object eventData)
```

PushRegistry:

- An application can register the inbound connection with an entry in application descriptor file or dynamically by calling `registerConnection`
- When an application is running it is responsible for all its I/O operations, if it is not running then the AMS listens for inbound notification requests and when a notification arrives it starts the MIDlet via `MIDlet.startApp`
- If all the static push connections defined in the descriptor file cannot be fulfilled during the installation, the user must be notified and the MIDlet suite must not be installed.
- Conditions where declaration cannot be fulfilled:
 - Syntax error in Push attributes
 - Port requested is already reserved by other application

- The declaration referencing MIDlet class that is not listed in MIDlet-<n> attributes of the application.
- Declaration of a protocol that is not supported
- Two MIDlet suites cannot be installed at the same time if they have a static push connection is common.
- If the implementation supports message buffering, it must be passed to the application once it is started.
- Not all protocols may support the Push capability, in which case ConnectionNotFoundException will be thrown by registerConnection and registerAlarm methods.
- The MIDlet is responsible for all the I/O operations from time of calling calling to Connector.open() to Connection.close()
- If the application closes the Connection, data loss can happen as the application closed the connection and the AMS is not listening for incoming notification requests. Only when the application is destroyed then the AMS starts back listening for incoming notification requests.
- By calling PushRegistry.registerConnection the applications informs the AMS that is target for inbound communication, even after the MIDlet is destroyed.
- If the application is delete from the phone then its dynamic connections are unregistered automatically.

RecordEnumeration:

- A call to nextRecord() or previous record returns a copy of the record
- If the RecordStore used by the RecordEnumeration is closed, the RecordEnumeration becomes invalid and all subsequent operations will throw RecordStoreNotOpenException or give invalid results, even if the same Recordstore is opened again later. In addition calls to hasNextElement() or hasPreviousElement will return false.

RecordStore:

- RecordStore locations are not exposed to the applications
- When a MIDlet is removed all the associated recordstores are also deleted.
- MIDlets are uniquely named using a unique name of the MIDlet suite plus the name of the recordstore. MIDlets are identified by the MIDlet-Vendor and MIDlet-Name attributes of the Application descriptor.
- RecordStore names are case sensitive.
- If the openRecordStore(String, Boolean, int Boolean) method is called by a MIDlet when the recordstore is already open a MIDlet in the same MIDlet suite, it returns the same object.

- The Authmode parameter passed to openRecordStore () is ignored if the record store already exists.
- Only the owning MIDlet suite can change the access mode of a Recordstore
- The closeRecordStore method must be called the same times as openRecordStore, only then the Recordstore will be actually closed. After closing a RecordStore all the listeners are removed, the RecordEnumeration becomes invalid, and if a MIDlet tries to access a closed RecordStore a RecordStoreNotOpenException will be thrown.

Spacer:

- Spacers cannot have commands and labels are always null and cannot be changed by the user. If the addCommand, setDefaultCommand or setLabel method is called it will throw an IllegalStateException

Sprite (Incomplete):

- The frames used to render a sprite are provided as a single mutable or immutable image. The image can be broken into equally-sized frames of a specified height and width
- Each frame is assigned a unique index number, with the image at the top-left corner being 0

Ticker:

- May contain line breaks

Timer:

- Timer class is thread safe,
- Timer does not offer real-time guarantees
- Timers function only within a single VM, and are cancelled when the VM exits. When the VM is started no Timers exist. They are created by the application's request.

ToneControl:

- If the setSequence() method is called when the player that the control belongs to is in either PREFETCHED or STARTED state, it will throw an IllegalStateException.
-