Mobile Information Device Programming (4)

Lecturer: Alireza Mousavi
School of Engineering & Design
www.brunel.ac.uk/~emstaam

MIDP Application Program Structure

Subjects:

- Application Execution Life Cycle
- MIDlet program Structure
- MIDlet State Model
- The MIDP UI component Model
- System and Application properties

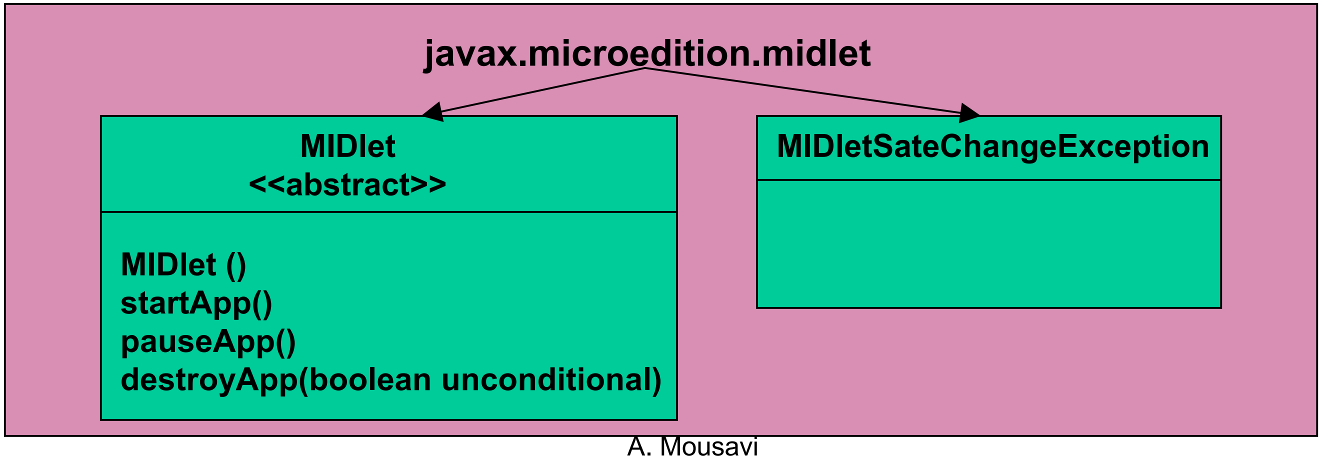
MIDlet Application Execution Life Cycle

MIDlets similar to Java applets implement methods (*life-Cycle methods)* that allows the Application Management Software (browser in case of applets) to control the behaviour of the MIDlet. They allow the AMS to control when a MIDlet *starts, stops and runs*

MIDlet Package

- The MIDlet Application life cycle is defined by MIDP 2.0
- The MIDP provides the MIDlet run-time life cycle definition and interactions with its run-time environment through:

  javax.microedition.midlet
- The *javax.microedition.midlet* defines two packages:



javax.microedition.midlet

MIDlet
<<abstract>>

MIDlet ()
startApp()
pauseApp()
destroyApp(boolean unconditional)

MIDletSateChangeException

MIDlet Life-Cycle Methods

The methods used by AMS to control MIDlet state are:

- Default Constructor: containing instance variable and objects

  Example: `public MyFirstMidlet ( ) {  … }`

- startApp( ): Activates MIDlet and obtains resources required

  ```
  public  void   startApp( ) { …}
  ```

- pauseApp( ): Releases the resources that are acquired by the MIDlet

  ```
  public  void   pauseApp( ) { … }
  ```

- destroyApp( ): Saves the state of the Midlet and releases the resources

  ```
  public  void   destroyApp( boolean unconditional) { … }
  ```

MIDlet Resources

A MIDlet creates, allocates and releases different types of resources during its life cycle these include:

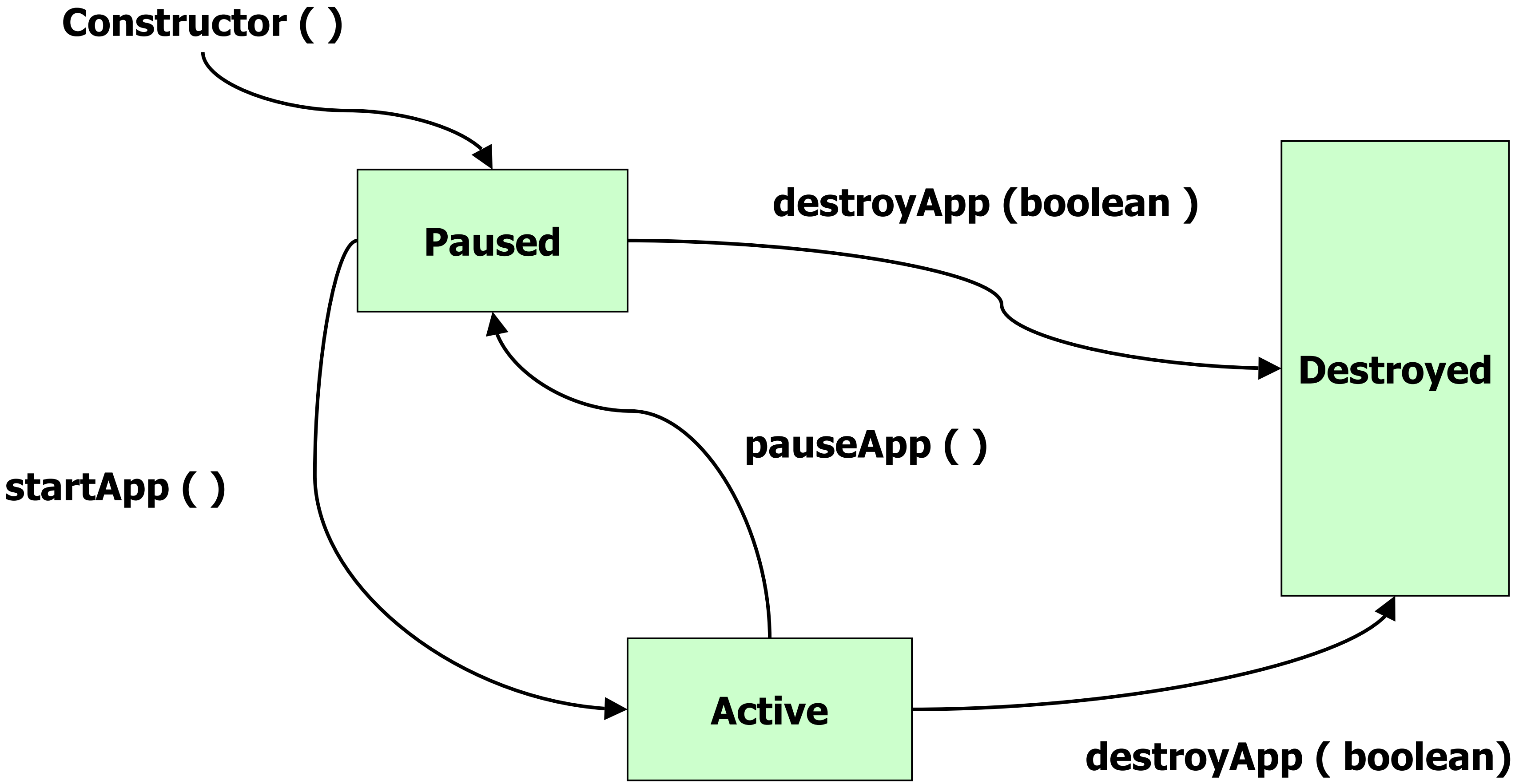
- Creates objects when needed and releases them when they are not needed – paused or terminated
- Reads (at startup) and stores (at shutdown) data from and to persistent storage
- Creating threads to perform concurrent tasks

The MIDlet State Model

A MIDlet can be in one of three states:

- **Paused:** When the AMS first creates a MIDlet, the MIDlet is in this state
- **Active:** The MIDlet is ready to run or running in this state
- **Destroyed:** The MIDlet terminates and can no longer move to any other state

Schematic view of Midlet State Model



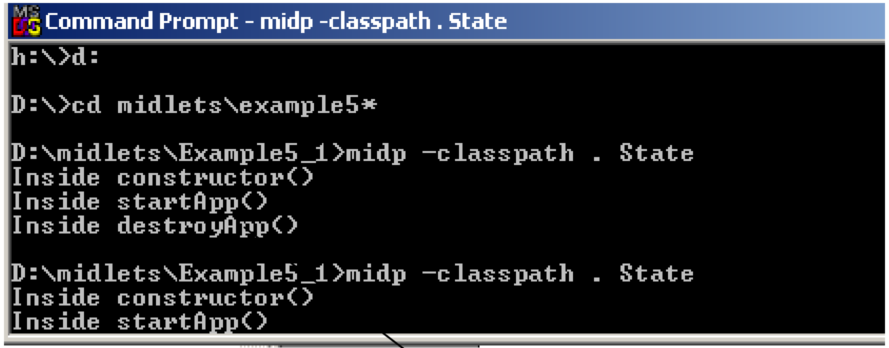
Exercise 1

Exercise 1: Investigating State Transitions:

1. Write the program save as State.java
2. Compile the program
3. Pre-verify the program
4. Run

Notice that the Paused method is never shown on the output why ?

Result 1



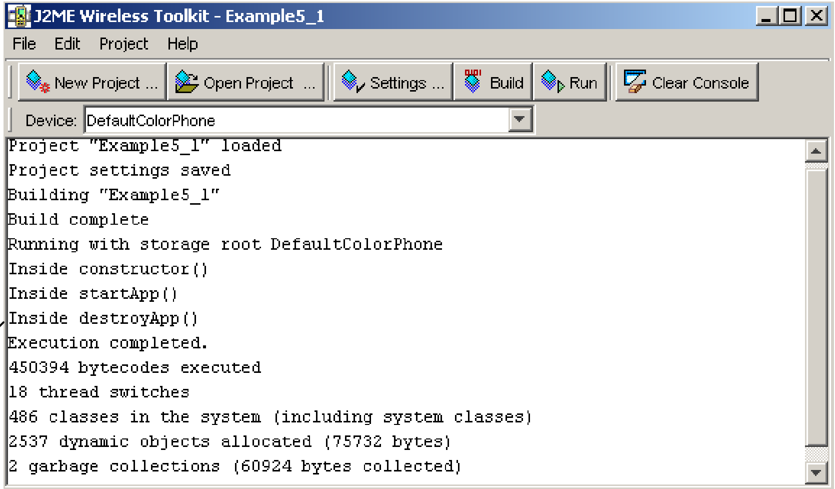
Command Prompt - midp -classpath . State

```
h:\>d:

D:\>cd midlets\example5*

D:\midlets\Example5_1>midp -classpath . State
Inside constructor()
Inside startApp()
Inside destroyApp()

D:\midlets\Example5_1>midp -classpath . State
Inside constructor()
Inside startApp()
```



J2ME Wireless Toolkit - Example5_1

Device: DefaultColorPhone

```
Project "Example5_1" loaded
Project settings saved
Building "Example5_1"
Build complete
Running with storage root DefaultColorPhone
Inside constructor()
Inside startApp()
Inside destroyApp()
Execution completed.
450394 bytecodes executed
18 thread switches
486 classes in the system (including system classes)
2537 dynamic objects allocated (75732 bytes)
2 garbage collections (60924 bytes collected)
```

Once you destroy Application

Exercise 2

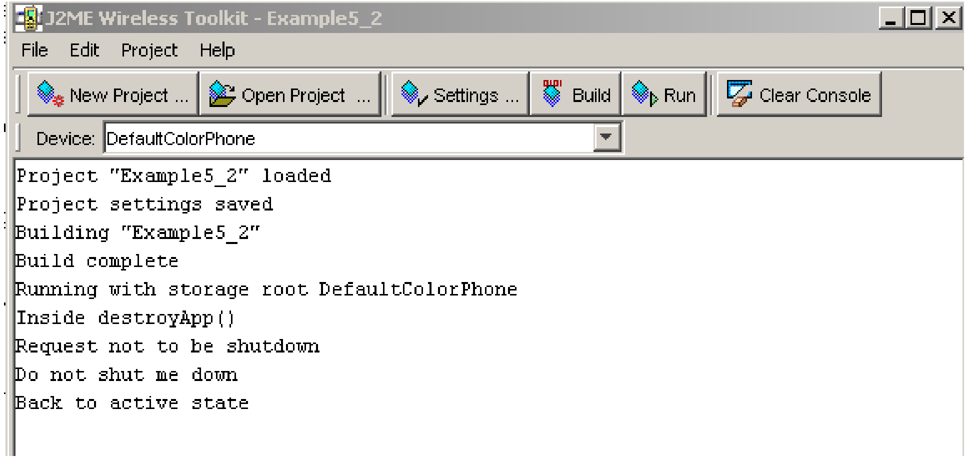
MIDletStateChangeException

1. Write the program save as ExceptTest.java
2. Compile the program
3. Pre-verify the program
4. Run

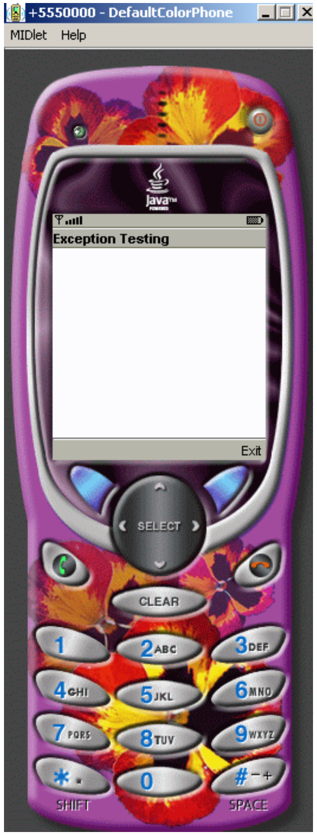
Notice that the first time we press Exit, we call *destroyApp(false);*

By passing in false we are informing the method that this is not an unconditional exit. Thus *destroyApp( )* will throw MIDletStateChangeException so we can not be shutdown

Result 2



```
Project "Example5_2" loaded
Project settings saved
Building "Example5_2"
Build complete
Running with storage root DefaultColorPhone
Inside destroyApp()
Request not to be shutdown
Do not shut me down
Back to active state
```

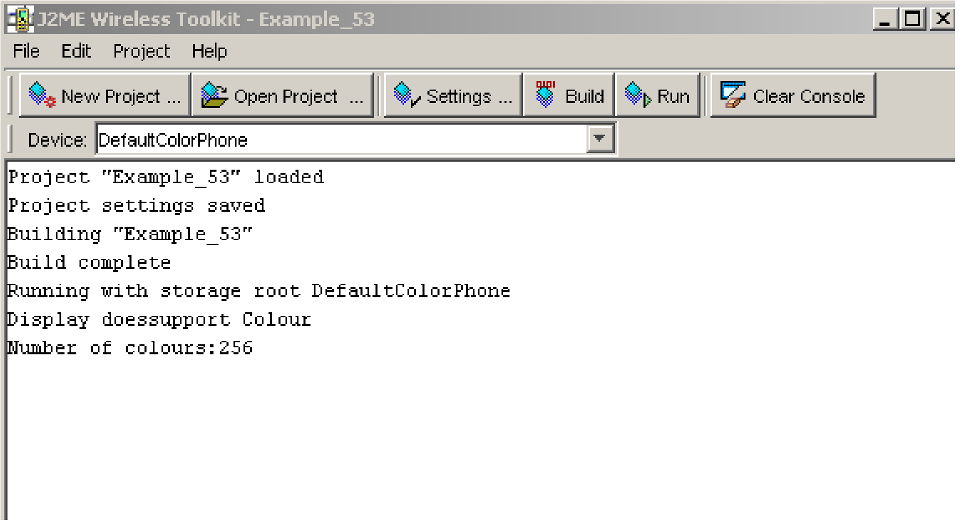


Exercise 3

Getting Device information with the help of Display, form, and Alert objects.

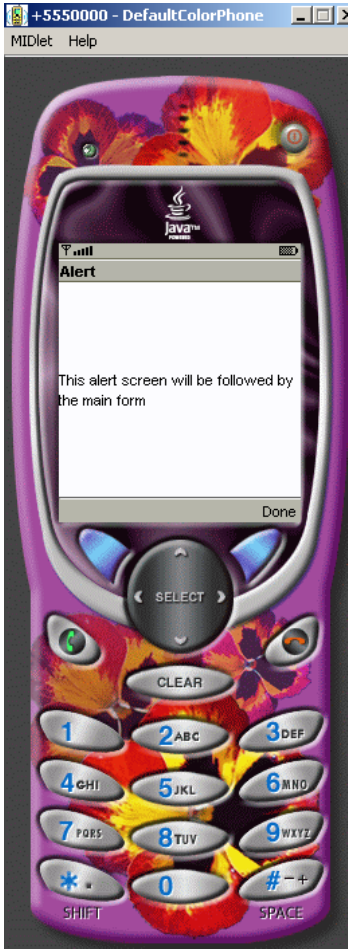
1. Write the program save as AlertTest.java
2. Compile the program
3. Pre-verify the program
4. Run

Result 3



Device: DefaultColorPhone

```
Project "Example_53" loaded
Project settings saved
Building "Example_53"
Build complete
Running with storage root DefaultColorPhone
Display doessupport Colour
Number of colours:256
```



This alert screen will be followed by the main form