***EXPERIMENT NO: 03***

***AIM:*** To study and implement J2ME programs

***THEORY:***

Around June,1999 at JavaOne Developer conference, sun has released the third standardization of Java in which J2ME brings the cross- platform functionality of the Java2, standard Edition (J2SE) to small device called as Java 2 platform, Micro Edition (J2ME) with a set of configurations and profiles, allowing mobile wireless device to share applications. J2ME was designed and released by Sun Microsystems (now subsidiary of Oracle) developed under the Java Community Process as Java Specification Request (JSR) 68, the platform replaced a similar technology Personal Java (It is a Java edition for mobile and embedded systems based on Java 1.1.8). This version aimed at embedded and resource constrained devices especially wireless devices. Java Micro Edition( formerly J2ME) design for embedded systems (such as Set- top boxes and Personal digital assistant (PDAs)), and Mobile devices( such as Smartphone, cell phones, and Personal Profile (framework for Java ME application, subset of Connected Device Configuration (CDC)). Java ME devices implement a Profiles, common of these are the Mobile Information Device Profile (MIDP) which include Mobile devices (such as such as Smartphone, cell phones, and Personal Profile) and Embedded systems (such as Set- top boxes and Personal digital assistant (PDAs)). Profiles are specific and subsets of Configurations, these are of currently two type, Connected Device Configuration (CDC) and the Connected Limited Device Configuration (CLDC).

### **J2ME or Java ME.**

Java ME was formerly known as Java 2 Platform, Micro Edition (J2ME). Both are primary components of Java platform or Java Micro Edition( formerly J2ME, Java ME), design for embedded systems and Mobile devices, all it means is Java for small devices. Java ME is a collection of configurations, profiles, and optional APIs packages that are designed for different parts of the small device (device that have less than 512 KB of memory) market.

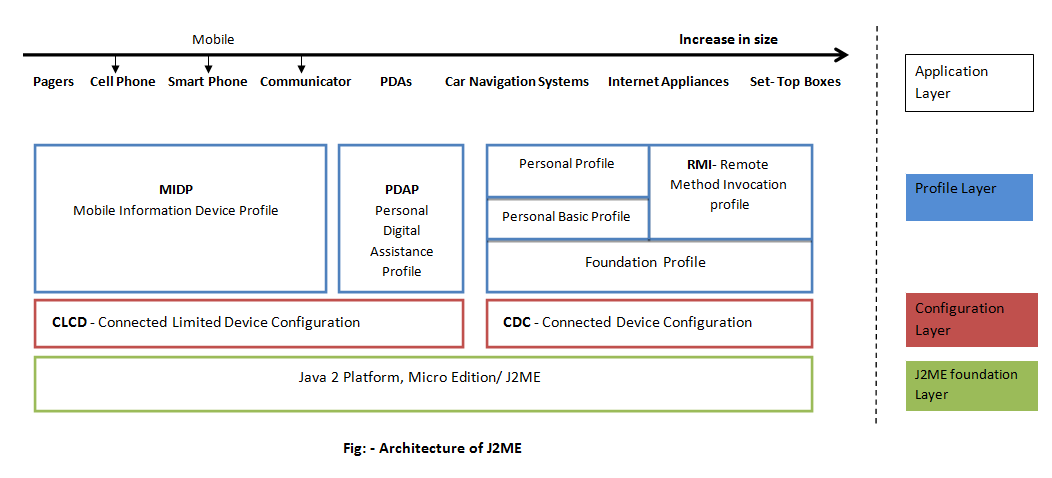
### **J2ME Technologies**

Around June,1999 at JavaOne Developer conference, sun has released the third standardization of Java in which J2ME brings the cross- platform functionality of the Java2, standard Edition (J2SE) to small device called as Java 2 platform, Micro Edition (J2ME) with a set of configurations and profiles, allowing mobile wireless device to share applications. Java 2 platform, Micro Edition(J2ME) is the second revolution in Java's history. J2ME isn't a piece of specification or software, while it is a collection of configurations, profiles, and optional packages that are designed for different parts of the small device (device that have less than 512 KB of memory) market. Small devices range in size from pagers, mobile phones, Smartphone, personal digital assistants (PDAs), and set-top boxes. J2ME is divided into Configurations, Profiles, and Optional APIs packages that are designed for different parts of the small device (device that have less than 512 KB of memory) market.

Configuration is designed for a specific kind of device based on memory constraints and processor power. A Configuration describes a set of APIs, Java Virtual Machine and library, but it does not capable itself for provide enough detail to developer for building complete applications. It also use subset of Java 2 Platform, Standard Edition(J2SE) APIs that will used on platform, as well as add additional APIs that may be necessary. Profile is more specific and subsets of configuration, a profile usually include a set of basic APIs for application life cycle, user interface, and persistent storage, to make a complete environment for building complex applications.Optional APIs define specific additional functionality that may be include in a particular configuration (or profile). The whole caboodle - profiles, configuration, and optional APIs that is implemented on a device is called stack.

**Architecture of J2ME**

J2ME uses Configurations, Profiles, and Optional APIs packages to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices, a profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices and a optional APIs package define specific additional functionality of core classes.



### **Generally, Architecture of J2ME define into four basic layer**

***Application Layer:*** Application layer targeted at two product groups, are:

Mobile, personal, and connected information devices such as Pagers, Mobile phones etc.

Fixed, shard, and connected information devices such as Car- Navigation Systems, Internet Appliances, Set - Top boxes etc.

***Profile Layer***:Profiles layer based on a configuration and additional API’s for user interface, persistent storage and factors necessary for developing wireless applications.

***Mobile Information Device (MIDP) Profile***: MIDP is specification published for the use of Java, mobile devices, and embedded system. It is a part of Java Micro Edition (Java ME) framework and top of a Connected Limited Device Configuration (CLDC). Its provides a standard platform for small, resource-limited, and wireless-connected mobile information devices. MIDP requires CLDC for implementation.

***Personal Digital Assistance (PDAP) Profile***:PDAP is specifically designed for Palmtop or similar devices with limited power, typical battery operated, user interface displays like a pointing device and character input, with minimum of 512kb (and maximum 16MB) combined ROM (read only memory) and RAM (random access memory) hardware.

***Personal Profile***:Personal profile specification repacks the Personal Java APIs Environment to provide J2ME specification for devices that need a high degree of Internet connectivity. This profile builds on top of foundation profile.

***Personal Basic Profile:***This profile is intended to provide basic Graphical User Interface (GUI) capabilities to devices where the high-fidelity, feature-rich Personal Profile is not fully utilized or necessary means which running the CDC and Foundation Profile.

***Remote Method Invocation (RMI) Profile:***RMI profile also builds on top of Foundation profile and supports inter-application RMI over TCP/IP connections for applications written to the Foundation Profile. It's provides the infrastructure to marshal objects as the parameters and return values of remote method calls while the dynamic class loading is utilized to make the marshalled objects available to a particular JVM during a remote call. A special type of protocol - wire protocol JRMP (Java Remote Method Protocol) is required for supported.

***Core Application Programming Interface (APIs) of RMI:***

**java.rmi**

***Foundation Profile:***The Foundation profiles specified for devices that can support a rich networked J2ME environment. It provides services for other device-specific profiles (such as Personal profile and RMI profile) to be layered on top of it. This profile specific design for CDC profiles that provide Graphical User Interface (GUI), Data storage, and distribution of java networking.

***Configuration Layer:***The configuration Layer is define the basic defines the basic run-time environment as a set of core classes and a specific JVM that run on specific kind of device based on memory constraints and processor power.

***Connected Limited Device Configuration (CLCD)***: - CLDC is a framework for ME defines the the basic set of APIs, library, and virtual-machine features for resource-constrained devices like pager, mobile phones, and mainstream Personal Digital Assistants (PDAs) and must be implemented into applications.

It provide a standard platform for developers for developing some complex applications on embedded devices with limited resource such as phones and pager by combined with one or more than one Profiles. CLDC is build on “scale down” version of Java virtual Machine (JVM) called KVM (Kilobytes Virtual Machine) for 16 - bit or 32- bit device with limited amounts of memory.

***Connected Device Configuration (CLCD):***CDC is a framework for Java ME defines the basic set of APIs, library, and virtual-machine features for resource-constrained devices like pager, mobile phones, and mainstream Personal Digital Assistants (PDAs) and must be implemented into applications. It provide a standard platform for developers for developing some complex application for embedded systems( from pager to set-top boxes) by combined with one or more than one Profiles.It specifies a full Java virtual Machine (JVM) called CVM (C Virtual Machine) and is used for 32- bit architectures requiring more than 2 MB of memory.

***J2ME Foundation Layer:***J2ME layer allow the flexibility of definition of APIs these are general purpose library, core classes, and independent to a particular device family. An abstraction of JSE network and I/O classes is designed for J2ME to make connection between mobile devices and web servers (if possible). It is called Generic Connection Framework (GCF) and used at the programming level.

### **Designing Goals of J2ME Architecture**

J2ME support much different architecture that J2SE and J2EE, hence design goals of J2ME architecture is also different from the Java alternate. Main important goals are:

Provide a strong and powerful architecture that can able to optimized form small size and space and run on very limited memory.

Architecture must be focused on devices that can be highly personalized, varied range of networking capabilities and services, often used by a single person.

Support and power up small memory devices with different capability. These small devices often varies in the area of requirements like memory, user interface, network connectivity, security, data storage, range and bandwidth.

Support maximum flexibility that provide maximum cross- platform capability of java language to support rapid changing marketplace and adapt to existing and unforeseen applications. Also, provide an optimized for delivering applications and data over a network connection.

Clean architecture of J2ME provide all resource for third party developers to develop an application of J2ME- supported devices independent of the Original Equipment Manufacturer (OEM). Also, support scale applications across devices with different capabilities, processing abilities and features

**Midlet In J2ME**

A **MIDlet** is a Java program for embedded devices, more specifically the Java ME virtual machine. Generally, these are games and applications that run on a cell phone A MIDlet requires a device that implements Java ME, MIDP to run. Like other Java programs, MIDlets have a “compile once, run anywhere” potential. MIDlet distributions also consist of a .jad file describing the contents of the JAR file. A MIDlet has to fulfill the following requirements in order to run on a mobile phone:

* The main class needs to be a subclass of javax.microedition.midlet.MIDlet
* The MIDlet needs to be packed inside a .jar file (e.g. by using the jar-tool)
* The .jar file needs to be pre-verified by using a preverifier.
* In some cases, the .jar file needs to be signed by the mobile phone’s carrier.

**MIDP Applications (MIDlets) and MIDlet Lifecycle**

The applications written for mobile information devices such as cellular phones and pagers are called MIDlets. Like applets, MIDlets are controlled by the software that runs them. In the case of an applet, the underlying software is a browser or the appletviewer tool and in the case of a MIDlet, the underlying software is the cell phone or two-way pager device implementation that supports the CLDC and MIDP. A MIDlet is a well-behaved MIDP application, which lives within the resource constraints, which runs and terminates when requested. All the devices, which support MIDP, are supposed to have a device-specific Application Management Software that takes care of installing, managing and removing MIDlets interactively.

#### **MIDlet lifecycle**

MIDlets move through a well-defined lifecycle consisting of five phases. It is the task of the Application Management Software to move MIDlets through these phases:

* Retrieval – The AMS retrieves the MIDlet from some source and reads the MIDlet into the device’s memory. The medium through which the MIDlet is downloaded depends on the device. It could be through a serial cable, an IRDA port, or a wireless network.
* Installation – Once the MIDlet is downloaded, the AMS installs the MIDlet on the device. During the installation process, the MIDP implementation verifies that the MIDlet does not violate the device’s security policies.
* Launching – A MIDlet is launched when a user selects it using the interface provided in the device. At this point, the MIDlet enters the KVM and the lifecycle methods of the MIDlet are invoked.
* Version Management – The AMS keeps track of all the MIDlets that are installed on the device including their version numbers. This information is used to upgrade a MIDlet to its new version.
* Removal – The AMS removes a MIDlet and cleans up the related resources from the memory.

A MIDlet can be in one of the three states after the Application Management Software launches it:

Paused A MIDlet enters the Paused state once it is created and initialized by the AMS. It can also enter this state when it is Active. Active this state means the MIDlet is running normally. A MIDlet goes to the Active state from the paused state if there are no runtime exceptions during its initialization. Destroyed this state means the MIDlet has released all its resources and is terminated. A MIDlet can reach this state either from the paused state due to a runtime exception during its initialization or from the active state when the user has chosen to close the application.

**MIDlet suites*:***

MIDlets are usually available through MIDlet suites. A MIDlet suite consists of two files, a .jar and a .jad file. The Java ARchive (JAR) file contains your compiled classes in a compressed and preverified fashion. All MIDlets must be preverified. This means that a checksum is computed, enabling the resource-constrained MID to easily check the integrity of the jar-file (using only a few hundred bytes of memory). Several MIDlets may be included in a MIDlet suite. Hence, the JAR file will contain all these MIDlet classes. The Java Application Descriptor (JAD) file is a plain text file containing information about your MIDlet suite. All MIDlets must be named in this file, the size of the JAR file must be included (and be correct!) and the URL to the JAR file must be present. In addition, the MIDlet suite version number is included here. This is essential information for a MID. If the suite is already installed, it will know if a newer version is available. The size of the JAR file is important information, the MID can determine if there is enough memory available to install the MIDlet suite. Every MIDLet contains 3 abstract methods that are called by the application manager to manage the life cycle of MIDLet. These abstract methods are startApp(), pauseApp(), destroyApp ( boolean unconditional)

public class Mymidlet extends MIDLet{

public void startApp(){

}

public void pauseApp(){

}

public void destroyApp( boolean unconditional ){

}}