PROGRAMMING CONCEPTS AND EMBEDDED PROGRAMMING IN C, C++ and JAVA:

Lesson-12: J2ME and JavaCard

Large JVM in Java

- Java byte codes that are generated need a larger memory when a method has more than 3 or 4 local variables. An embedded Java system may need a minimum of 512 kB ROM and 512 kB RAM because of the need to first install JVM and then run the application.

J2ME (Java 2 Micro Edition) and JavaCard has small size virtual machine

- Use of J2ME (Java 2 Micro Edition) or Java Card or EmbeddedJava helps in reducing the code size to 8 kB for the usual applications like smart card

J2ME

- Java 2 Micro Edition
- Provides Configurations and profiles for the small devices

J2ME new virtual machine, KVM (Kilobytes Virtual Machine)

- KVM as an alternative to JVM.
- When using the KVM, the system needs a 64 kB instead of 512 kB run time environment.

KVM features

- Use of the following data types is optional. **(*i*)** Multidimensional arrays, **(*ii*)** long 64-bit integer and **(*iii*)** floating points.
- Errors are handled by the program classes, which inherit only a few needed error handling classes from the java I/O package for the Exceptions.

KVM features

- Use of a separate set of APIs (application program interfaces) instead of JINI. JINI is portable. But in the embedded system, the ROM has the application already ported and the user does not change it.

KVM features

- There is no verification of the classes. KVM presumes the classes as already validated.
- There is no object finalization. The garbage collector does not have to do time consuming changes in the object for finalization
- The class loader is not available to the user program. The KVM provides the loader

KVM features

- Thread groups are not available.
- There is no use of java.lang.reflection. Thus, there are no interfaces that do the object serialization, debugging and profiling

J2ME features

- Use core classes only.
- Classes for basic run time environment form the VM internal format and only the programmer's new Java classes are not in internal format
- Provide for configuring the run time environment.

Configurability

- Examples of configuring are deleting the exception handling classes, user defined class loaders, file classes, AWT classes, synchronized threads, thread groups, multidimensional arrays, and long and floating data types.
- Other configuring examples are adding the specific classes for connections when needed, data-grams, input output and streams

J2ME

- Create one object at a time when running the multiple threads.

J2ME

- J2ME provides the optimised run-time environment.
- Instead of the use of packages, J2ME provides for the codes for the core classes only. These codes are stored at the ROM of the embedded system.

J2ME programming using small code size

- Reuse the objects instead of using a larger number of objects.
- Use scalar types only as long as feasible

J2ME

- Provides for configuring the run time environment. Examples of configuring are deleting the exception handling classes, user defined class loaders, file classes, AWT classes, synchronized threads, thread groups, multidimensional arrays, and long and floating data types.

J2ME

- J2ME provides Scaleable OS feature
- J2ME need not be restricted to configure the JVM to limit the classes.
- Configuration augmentation by Profiler classes. For example, MIDP (Mobile Information Device Profiler). A profile defines the support of Java to a device family.

Mobile information device profile (MIDP)

- MIDP used as in PDAs, mobile phones and pagers.
- A touch screen or keypad.
- A minimum of 96 x 54 pixel color or monochrome display.
- Wireless networking.
- A minimum of 32 kB RAM, 8 kB EEPROM or flash for data and 128 kB ROM.

MIDP

- MIDP classes describe the displaying text.
- Profile Classes for the network connectivity. For example, for HTTP. [Internet Hyper Text Transfer Protocol.]
- Provides support for small databases stored in EEPROM or flash memory.
- Schedules the applications and supports the timers.

J2ME Two alternative configurations

- Connected Device Configuration (CDC) and Connected Limited Device Configurations (CLDC).
- CDC inherits a few classes from packages for *net*, *security*, *io*, *reflect*, *security.cert*, *text*, *text.resources*, *util*, *jar* and *zip*.

J2ME Two alternative configurations

- CLDC does not provide for the applets, awt, beans, math, net, rmi, security and sql and text packages in java.lang. There is a separate javax.microedition.io package in CLDC configuration. A PDA (personal digital assistant) uses CDC or CLDC.

Java Card or EmbeddedJava

- Use of or Java Card or EmbeddedJava helps in reducing the code size to 8 kB for the usual applications like smart card. How? The following are the methods.

JavaCard

- Reducing up to the code size to 8 kB possible for the usual applications like smart card.

JavaCard

- The Java advantage of platform independence in byte codes is an asset
- The smart card connects to a remote server.
- The card stores the user account past balance and user details for the remote server information in an encrypted format. It deciphers and communicates to the server the user needs after identifying and certifying the user.

JavaCard

- The intensive codes for the complex application run at the server.
- A *restricted run time environment exists in Java classes* for connections, data-grams, character-input output and streams, security and cryptography only

Summary

We learnt J2ME provides

- benefits of small JVM
- optimized run-time environment
- configurations and profiles in J2ME for small devices
- Configuration augmentation
- extensive class libraries for network and web applications
- modularity, robustness

We learnt J2ME provides

- secure restricted permissions
- portability
- platform independence for programming small devices such as mobile and card

End of Lesson 12 of Chapter 5