CS 4/55231
Internet Engineering

Kent State University
Dept. of Computer Science

LECT-13

Active Document & Mobile Code Technology

Principal Challenges

- Active Elements
- Universal Portability
- Security

Mobile Code Technology & Java

Mobile Code
=
Server/ Browser
+
Java Language
+
Java Virtual Machine

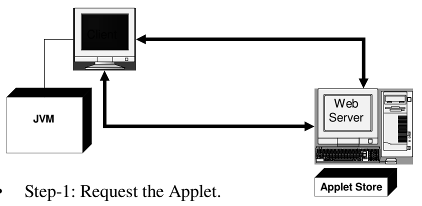
Cross Platform Operability
Execution Environment
Code Distribution
Safety

Java Technology

- Java technology takes the concept of "sharing" via Internet a step ahead, sharing active programs.
- Active programs known as applets. can be embedded inside HTML pages. Applets are transparently downloaded into Browser along with the HTML pages in which they appear.
- Applets behave the same way regardless of where they come from, or what kind of machine they are being loaded into and run on.

Java is a way of distributing software via Internet.

Active Document Model



- Step-1: Request the Applet.
- Step-2: Receive the Applet.
- Step-3: Load and Start the Applet.
- Step-4: Applet performs active display on Client.
- Step-5: Discard the Applet.

Java Virtual Machine

INTERNET ENGINEERING

- The Java Virtual Machine is an abstract computing machine.
  - Like a real computing machine, it has an instruction set and uses various memory areas.
  - JVM does not assume any particular implementation technology or host platform.
  - JVM can be implemented directly in silicon.
- JVM does not assume that the instructions it executes were generated from a Java source code.

Java Virtual Machine

INTERNET ENGINEERING

- The Java Virtual Machine code is written in the informal "virtual machine assembly language".
- The instruction set for JVM has been designed after RISC processor model.
- JVM opcodes are one byte long. known as “bytecodes”. The 1-byte size also means:
  - The compiled code being very compact.
  - Instruction set must stay small.
- If a platform has JVM installed it can run a Java application directly.
- Bytecodes when ‘dressed’ as “Applets” can be run from a JVM equipped Web Browser.

The magic of Bytecodes

INTERNET ENGINEERING

- A Java Applet is a portable unit of mobile code. Java achieves portability by compiling applets to JVM.
- Byte codes brings the compiled instructions to the lowest level possible without making them machine dependant.
- Bytecodes make Java a partially compiles language. It is about 80% compiled and 20% interpreted.
- JVM knows nothing of the Java programming language. JVM only understands a special file called class file, which contains the bytecodes.

Still, Applets run about 15 time slower than a compiled code!

JVM Code (Class File)

INTERNET ENGINEERING

```
ClassFile {

    u4 magic;
    u2 minor_version;
    u2 major_version;
    u2 constant_pool_count;
    cp_info constant_pool[constant_pool_count-1];
    u2 access_flags;
    u2 this_class;
    u2 super_class;
    u2 interfaces_count;
    u2 interfaces[interfaces_count];
    u2 fields_count;
    field_info fields[fields_count];
    u2 methods_count;
    method_info methods[methods_count];
    u2 attributes_count;
    attribute_info attributes[attributes_count];
}
```

How JVM Works?

INTERNET ENGINEERING

- A new frame is created each time a Java method is invoked, and with it is created a new operand stack and set of local variables for use by that method.
- JVM is stack-oriented, with most operations taking one or more operands from the operand stack of the Java Virtual Machine's current frame, or pushing results back onto the operand stack.
- Active data space: At any one point of the computation, there are thus likely to be many frames and equally many operand stacks per thread of control, corresponding to many nested method invocations. Only the operand stack in the current frame is active.
- Argument Passing: If n arguments are passed to a Java instance method, they are received in the local variables numbered 1 through n of the newly created frame. The arguments are received in the order they were passed.

JVM Synchronization

INTERNET ENGINEERING

- The Java Virtual Machine provides explicit support for synchronization through its monitorenter and monitorexit instructions.
- During the time the executing thread owns the monitor, no other thread may acquire it. If an exception is thrown during invocation of the synchronized method, and the synchronized method does not handle the exception, the monitor for the method is automatically released before the exception is rethrown out of the synchronized method.
- The monitorenter and monitorexit instructions exist to support Java's synchronized statements. A synchronized statement acquires a monitor on behalf of the executing thread, executes the body of the statement, then releases the monitor:

Example of Bytecode

INTERNET ENGINEERING

```
void spin() {
int i;
 for (i = 0; i < 100; i++) {
            ; // Loop body is empty
            }
}
```

```
void spin() {
Method void spin()
 0   iconst_0      // Push int constant 0
 1   istore_1      // Store into local 1 (i=0)
 2   goto 8        // First time through don'tincrement
 5   iinc 1 1      // Increment local 1 by 1 (i++)
 8   iload_1       // Push local 1 (i)
 9   bipush 100    // Push int constant (100)
 11 if_icmplt 5    // Compare, loop if < (i < 100)
 14 return         // Return void when done
```

Example of Stack Operation

INTERNET ENGINEERING

```
int align2grain(int i, int grain) {

 return ((i + grain-1) & ~(grain-1));
}
```

```
5 iload_2          // Load grain onto operand stack
6 iconst_1         // Load constant 1 onto operand stack
7 isub             // Subtract; push result onto stack
8 iconst_m1        // Load constant -1 onto operand stack
9 ixor             // Do XOR; push result onto stack
```

- First grain-1 is calculated using the contents of local variable 2 and an immediate int value 1.
- These operands are popped from the operand stack and their difference pushed back onto the operand stack, where it is immediately available for use as one operand of the ixor instruction (recall that ~x == -1^x).
- Similarly, the result of the ixor instruction becomes an operand for the subsequent iand instruction.

JVM Verification

INTERNET ENGINEERING

- Web browser download already-compiled class files. Before passing on to JVM, bytecodes are verified through extensive scrutiny.
- Besides security it also improves performance. For example, the JVM will already know the following:
  - There are no operand stack overflows or underflows.
  - All local variable uses and stores are valid.
  - The arguments to all the Java Virtual Machine instructions are of valid types.
- 4 passes are made during verification.

JVM Verification-1

INTERNET ENGINEERING

- Pass 1: When a prospective class file is loaded, JVM ensures that the file has the basic format of a Java class file.
- Examples:
  - The first four bytes must contain the right magic number.
  - All recognized attributes must be of the proper length.
  - The class file must not be truncated or have extra bytes at the end.
  - The constant pool must not contain any superficially unrecognizable information.

JVM Verification-2

INTERNET ENGINEERING

- Pass 2: When the class file is linked, the verifier performs all additional verification that can be done without looking at the code array of the Code attribute.
- Examples:
  - Ensuring that final classes are not sub-classed, and that final methods are not overridden.
  - Checking that every class (except Object) has a super-class.
  - Checking that all field references and method references in the constant pool have valid names, valid classes, and a valid type descriptor.

JVM Verification-3

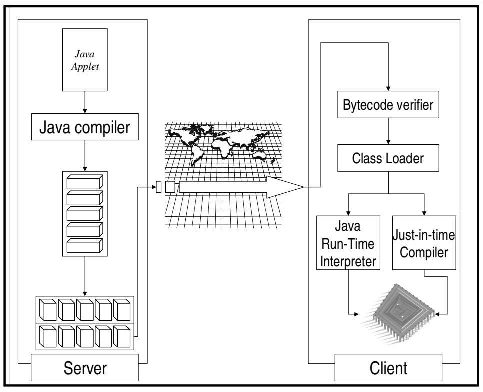
INTERNET ENGINEERING

- Pass 3: Still during linking, the verifier checks the code array of the code attribute for each method of the class file by performing data-flow analysis on each method.
- The verifier ensures that at any given point in the program, no matter what code path is taken to reach that point:
  - The operand stack is always the same size and contains the same types of objects.
  - No local variable is accessed unless it is known to contain a value of an appropriate type.
  - Methods are invoked with the appropriate arguments.
  - Fields are assigned only using values of appropriate types.
  - All opcodes have appropriate type arguments on the operand stack and in the local variables.
- Most expensive pass!

JVM Verification-4

- Pass 4: For efficiency reasons, certain tests that could in principle be performed in Pass 3 are delayed until the first time the code for the method is actually invoked. In so doing, Pass 3 of the verifier avoids loading class files unless it has to.

INTERNET ENGINEERING



INTERNET ENGINEERING

JVM Defense

- JAVA Language Defense:
  - memory layout is not decided by compiler but by loader at runtime.
  - JAVA does not support "pointers", instead memory is referenced via names which are resolved at run time by interpreter.
- Verifier Defense:
- Class Loader Defense:
  - The namespace is local to a class. A class can access objects that are within its namespace.
- Access Control Defense:
  - File access parameters can be separately specified for imported codes.
- Applet Certification:

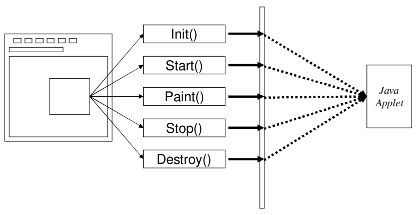
INTERNET ENGINEERING

Web Browser and Applets

- Java Applets vs. Java Applications: A Java applet is not a complete Java application. It is a component which can run by a Web Browser.
- An Web Browser: Controls the full lifecycle of an Applet. It supplies the attributes specified in HTML tags to applet and it also provides the execution environment and the main function.

INTERNET ENGINEERING

Applet Life Cycle Management



INTERNET ENGINEERING

HTML Applet Tag

Example of Java Applets

```
<APPLET
CODE=Hello.class
CODEBASE="applet/myapps"
WIDTH=300
HEIGHT=200 ALLIGN=left>
<PARAM NAME=param1 VALUE="Java is Cool">
<PARAM NAME=param2 VALUE="Good Bye">
</APPLET>
```

INTERNET ENGINEERING