ComS 587X Fall 2002
Lecture 5

InetAddress Class

- Provides methods for obtaining IP addresses and host names
  - boolean b = ip1.equals(ip2)
    - Compares IP addresses ip1 and ip2 for equivalence
  - byte bytes[] = ip.GetAddress()
    - Returns IP address as an array of bytes
  - InetAddress ips[] = InetAddress.getAllByName("hostname")
    - Returns array of InetAddress

InetAddress Class

- InetAddress ip = InetAddress.getByName("hostname")
  - Returns a single IP address
- String s = ip.GetHostAddress()
  - Returns dotted-decimal string IP address
- InetAddress ip = InetAddress.getLocalHost()
  - Returns an IP address for the local host
- String s = ip.getHostName()
  - Returns the name for the given IP

InetAddress Class

- boolean b = ip.isMulticastAddress()
  - Returns true if the IP address is a class D address
- String s = ip.toString()
  - Returns a string representation of the IP address object

Streams

- Streams are used in Java to perform byte I/O
- Read/write bytes from/to
  - Byte arrays
  - Files
  - Pipes
  - StringBuffers
  - Network sockets
- Streams are unidirectional

InputStream Methods

- All InputStreams must implement these methods:
  - int i = is.Available();
  - is.close();
  - is.mark();
  - boolean b = is.markSupported();
  - int byte = is.read();
  - int bytesread = is.read(byte[] bytes);
  - int bytesread = is.read(byte[] bytes, offset, length);
  - is.reset();
  - long bytesSkipped = is.skip(length);

InputStreams include:

- InputStream is = new FileInputStream(“pathname”);
  - Opens “pathname” for reading
- byte[] b = {0, 1, 2, 3, 4};
- InputStream is = new ByteInputStream(b);
  - Allows reading the bytes from the array b
- StringBuffer sb = new StringBuffer(“01234”);
- InputStream is = new StringBufferInputStream(sb);
  - Allows reading from the StringBuffer sb
- System.in
  - Special InputStream provided by the Java VM

OutputStream methods

- All OutputStreams must implement the following methods:
  - os.close();
  - os.flush();
  - os.write(byte b);
  - os.write(byte[] bytes);
  - os.write(byte[] bytes, offset, length);

OutputStreams include:

- OutputStream os = new FileOutputStream(“pathname”);
  - Opens/truncates “pathname” for writing
- byte[1024] b;
- OutputStream os = new ByteOutputStream(b);
  - Allows writing bytes to the array b
- StringBuffer b = new StringBuffer();
- OutputStream os = new StringBufferOutputStream(sb);
  - Allows writing to the StringBuffer sb
- System.out, System.err
  - Special OutputStreams/PrintStreams provided by the Java VM

FilterStreams

- Provide buffering and access to data types other than bytes
- Input:
  - bis = new BufferedInputStream(is);
    - Provides input buffering w/o additional methods
  - dis = new DataInputStream(is);
    - Add methods to read primitive types
  - pbis = new PushBackInputStream(is);
    - Adds unread() method to push bytes back into stream

FilterStreams (cont)

- Output:
  - bos = new BufferedOutputStream(os);
    - Provides output buffering w/o additional methods
  - dos = new DataOutputStream(os);
    - Adds methods to write primitive types
  - ps = new PrintStream(os);
    - Adds print() method for primitive types, println() method for printing primitive types with a following newline, setError() method, and checkError() method

Readers and Writers

- Support Unicode (2-byte and variable width) characters
- All Readers must implement these methods:
  - boolean b = r.ready();
  - r.close();
  - r.mark(); boolean b = r.markSupported();
  - int c = r.read();
  - int charsread = r.read(byte[] bytes);
  - int charsread = r.read(byte[] bytes, offset, length);
  - r.reset();
  - long bytesSkipped = r.skip(length);

Readers

- Reader r = new FileReader(“pathname”);
  - Opens “pathname” for reading
- char[] c = {'a', 'b', 'c', 'd', 'e'};
- Reader r = new CharArrayReader(c);
  - Allows reading the chars from the array c
- String s = “01234”; Reader r = new StringReader(s);
  - Allows reading from the String s
- Reader r = new InputStreamReader(is);
  - Creates a Reader from an InputStream
- Reader br = new BufferedReader(r);
  - Adds buffering to Reader

Writer Methods

- All Writers must implement the following methods:
  - w.close();
  - w.flush();
  - w.write(int c);
  - w.write(char[] chars);
  - w.write(String s);
  - w.write(char[] chars, offset, length);

Writers

- Writer w = new FileWriter(“pathname”);
  - Opens/truncates “pathname” for writing
- Writer w = new CharArrayWriter(c);
  - char[] c = w.toCharArray();
  - Allows writing chars to an array
- Writer w = new StringWriter(sb);
  - StringBuffer sb = w.getBuffer();
  - Allows writing to a StringBuffer
- Writer w = new OutputStreamWriter(os);
  - Allows making an OutputStream into a Writer
- Writer bw = new BufferedWriter(w);

Object Persistence

- Objects may be serialized into an array of bytes
- Objects may be reconstructed (deserialized) from an array of bytes
- Serialized objects are *persistent*
- Java classes that implement Serializable may automatically be serialized
  - All member vars must also be Serializable
- Objects are assigned an SUID to prevent version problems

How Object Persistence Works

- Java VM automatically provides serialization for classes marked as implementing *java.io.Serializable*
  - Provides the serialize() method for the class
  - Class must provide a default constructor (no args)
- Members are recursively serialized
  - Except static members and members marked with the *transient* keyword
- ObjectInputStream (extends DataInputStream) method readObject() to read objects
- ObjectOutputStream (extends DataOutputStream) method writeObject() to write objects

Summary

- Input and output streams
- Input and output filters
- Readers and Writers
- Object serialization