1. Topic
    1. Basics of j2me networked applications
    2. not games
2. No powerpoint
    1. can feel intelligence sapping away.
3. Who are you
    1. Who has cell phones
    2. who has used threads in a non trivial manner?
    3. non java dev
    4. on client(swing, applet)
    5. On server
        1. enterprise—ejb, jms
        2. web—servlet, jsp
    6. j2me
4. Who am I
    1. Using java (off and on) since 1998
        1. applets, trivial client apps, web stuff
    2. On team that build one J2ME application (MIDP 1.0), been working with it for about 6 months. Deployed to real cell phones (if not a lot)
5. Nature of problem
    1. Non technical user
        1. Used to consumer appliance of phone—stuff mostly works, isn't complex
    2. Distributed access to database
        1. Querys—readonly
        2. Architecture
            1. Picture
    3. Interface—phone
        1. picture
        2. what do you have—12 number/letter buttons, 2 soft buttons.
        3. Like swing, create forms (jpanel), add elements, laid out top to bottom, mostly.
6. J2ME basics
    1. Not just cell phones, pdas, other limited hardware.
    2. Intersection with j2se—java.io, java.lang, java.util packages, plus javax.microedition, which is UI, lifecycle, io and some other good stuff
        1. But, everything in j2me behaves like j2se
    3. Picture
    4. Midlet—abstract class
7. Specifications
    1. JSR 37, 118 (MIDP), JSR 30, 139 (CLDC)
    2. CLDC underneath MIDP, others
    3. different versions of specs
        1. MIDP 1.0 fairly broad support
        2. MIDP 2.0 released in nove 2002, implementations still sparse.
        3. CLDC is device configuration, MIDP sits on top, handles ui, higher abstractions
    4. Resource constrained; spec says
        1. 96x54 screen size
        2. 8kb of non volatile memory (aka, disk space); available as byte array

    3. 32kb of heap size
    4. No floating point support (changed in CLDC 1.1, or use MathFP)
8. Phone considerations
    1. Note that most phones have are better off than spec—t720 has 512kb of heap
    2. Choose phone(s) to support and test
        1. Like applets—write once, debug everywhere.
        2. Two players in the phone situation: manufacturers (Nokia) and service providers (AT&T Wireless). Manufacturer implements the spec, provides other functionality, but service provider has final say on J2EE—verizon doesn't support it, even though their phones do.
        3. Can't just 'install a new JDK'
    3. Optional packages—JDBC, RMI, LBS
        1. Have to be implemented by manufacturer
    4. Other proprietary extensions--muglets
9. Why J2ME rather than WML?
    1. Common skillset
    2. Rich client
        1. Why use swing over html
        2. can build state through series of screens, then hit network once.
        3. Richer GUI, can dynamically change based on client conditions.
            1. 100 items vs 5 items
        4. Optional packages can lead to richer functionality
10. Emulators and real phones
    1. Why use an emulator?
        1. Speed of edit-compile-test
        2. Expense
            1. Most phones, can only download from network (via .jad file)
            2. Motorola/CodeWarrior can download directly from PC
    2. Most windows only.
        1. Some manufacturers provide
        2. Some service providers provide
        3. Direct experience
            1. Sun WTK
            2. WSDD
    3. Not perfect
        1. Faster network connection
        2. Faster data input
        3. Faster disk
11. J2EE
    1. Struts application
    2. Talked to remote database
    3. Generated XML via JSP
    4. Didn't use action forms
        1. provide two thing, we didn't need either.
            1. automatic error handling
            2. VO on server side, but we're doing relatively simple query strings
12. What kind of data

1. From client to server, get and post with params as strings
2. From Server to Client, text and pictures

13. Options for data transfer
    1. Pictures—options limited—array of bytes
        1. Sent via servlet
        2. PNG support in spec, some support other formats.
        3. JAI on server lowered resolution of photos—not priority.
        4. Can turn array of bytes into image, not vice versa.
    2. Text
        1. Really marshalled VOs
        2. Own proprietary format
            1. Sockets
            2. HTTP
        3. Not all support sockets, brittle with changes, hard work defining
        4. SOAP
        5. Didn't understand soap.
            1. Now have biz logic in action classes—oh no!
        6. RMI
            1. Optional package—would limit supported phones.
        7. XML over HTTP
            1. All support
            2. Easy to debug, perf test
            3. Can be used for different clients
            4. Looking at SOAP

14. Identifying users
    1. HTTP is, as ever, stateless
    2. Authentication
        1. No container managed auth
        2. Roll your own
            1. HTTP basic auth, form
        3. Built own form
    3. Sessions
        1. Cookies not supported automatically
        2. Tokens in query parameter
            1. Could have gone with cookies, but were sending params anyway.

15. Network issues
    1. Slow—AT&T wireless says old network optimum is 40Kb/sec—not bad for modem!
        1. We don't see. Takes 3-5 minutes to download 70KB app
        2. Varies by carrier; 15-20 sec on sprint
    2. Unreliable
        1. Cache data
            1. size
            2. security
            3. stale
        2. Catch IOException when network unavailabe
        3. User used to cell phones fading in and out.

        1. Finally, something where consumer appliance nature helps
    3. Number of connections can be limited.
        1. Device dependent, not part of spec (that I could find)
16. Threading issues
    1. Again, how many really understand threading
        1. I don't—concurrency scares me.
    2. All methods on the system thread should return immediately(GUI)
        1. we use for network access, should use for record store access.
    3. Picture
    4. Actually locked up one phone when network call from system thread didn't return
        1. asking permission (not part of spec that I could find), but couldn't answer yes because system thread was waiting for network call.
    5. Callback
17. Security
    1. Over wire, https is supported by some phones
    2. On phone, username and password could be encrypted
        1. Covered (excruciatingly) in EntJ2ME
    3. Not needed for our app.
18. Performance Issues
    1. Network limiting factor
    2. Profiler in WTK
        1. For seeing memory usage, object creation.
    3. Did perf testing of J2EE
        1. Can support at lest 10 concurrent users with 6 second response time.
        2. Lower limit, since j2me will not request as fast.
19. UI Issues
    1. Ease of use paramount
        1. Remember, it's an appliance.
    2. Balance between screen size (showing lots of data) and network hits (getting dat often) and local caching (storing local data)
    3. Entering letters sucks
        1. Cache where can
        2. specify inputs as number or letter
        3. Use interweb
            1. Picture
    4. Rich client helps
        1. If there's only one choice, don't make them choose it, give it to them.
            1. Pagination example
    5. Lack of control—can't move buttons around—like a browser.
        1. Differs between phones—Motorola enter text inplace, samsung enter text in popup box
        2. Phone calls interupt, pause app.
    6. Testing important.
20. Resources
    1. All online at http://www.mooreds.com/j2me/
    2. Favs
        1. javaranch forum

    2. EnterpriseJ2ME—sampler tray of j2me technologies
    3. java.sun.com/j2me
        1. specs, tips
    4. Service providers website
    5. Phone manufacturers website
    6. Style Guide
21. Conclusion
    1. A lot like internet in 97
        1. Lots of promise
        2. Lots of diversity
        3. Lots of roll your own.
22. Thanks
    1. John Argo, Brian Rook for reviewing presentation and very helpful suggestions
    2. Brian Rook for letting me help build this application
    3. BJUG and you!
23. Questions

Resources:
EnterpriseJ2ME Book, Yuan
MIDP homepage:
http://java.sun.com/products/midp/index.jsp
Powerpoint bad:
http://www.edwardtufte.com/tufte/powerpoint
J2ME device database:
http://kissen.cs.uni-dortmund.de:8080/devicedb/index.html
J2ME weblog:
http://jroller.com/page/shareme
Another J2ME weblog:
http://www.enterprisej2me.com/blog/java/
kXML, kSOAP:
http://kxml.enhydra.org/, http://ksoap.enhydra.org/
Article about session tracking:
http://www.javaworld.com/javaworld/jw-04-2002/jw-0426-wireless.html
HTTP Connections and background threads:
http://developers.sun.com/techtopics/mobility/midp/ttips/httpthrds/
J2ME Emulators:
http://www.jroller.com/page/shareme/J2MEEmulators
J2ME style guide:
http://java.sun.com/j2me/docs/alt-html/midp-style-guide7/
JavaRanch J2ME Forum:
http://saloon.javaranch.com/cgi-bin/ubb/ultimatebb.cgi?ubb=forum&f=41
AT&T:
http://www.attwireless.com/developer/
Sprint:
http://developer.sprintpcs.com/
T-Mobile:
http://developer.t-mobile.com/tmobile/

Nextel:
http://developer.nextel.com/portal/index.jsp
Microjava.com, device database:
http://www.microjava.com/
Optional package listing:
http://developers.sun.com/techtopics/mobility/apis/