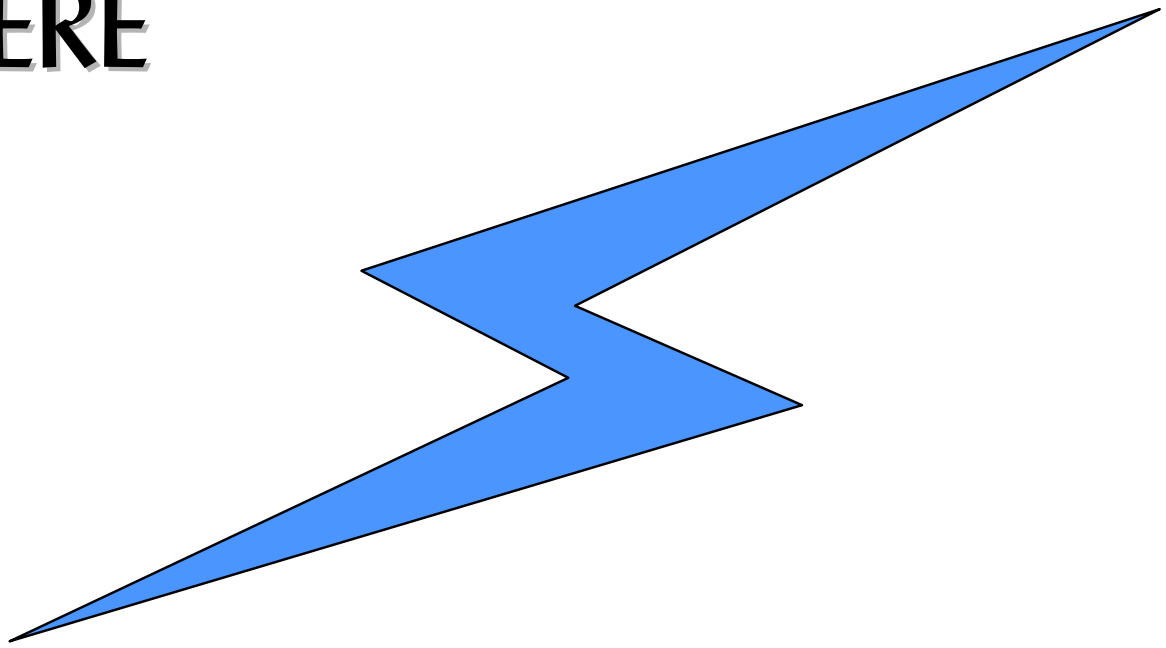
WEBSPHERE



JAVA

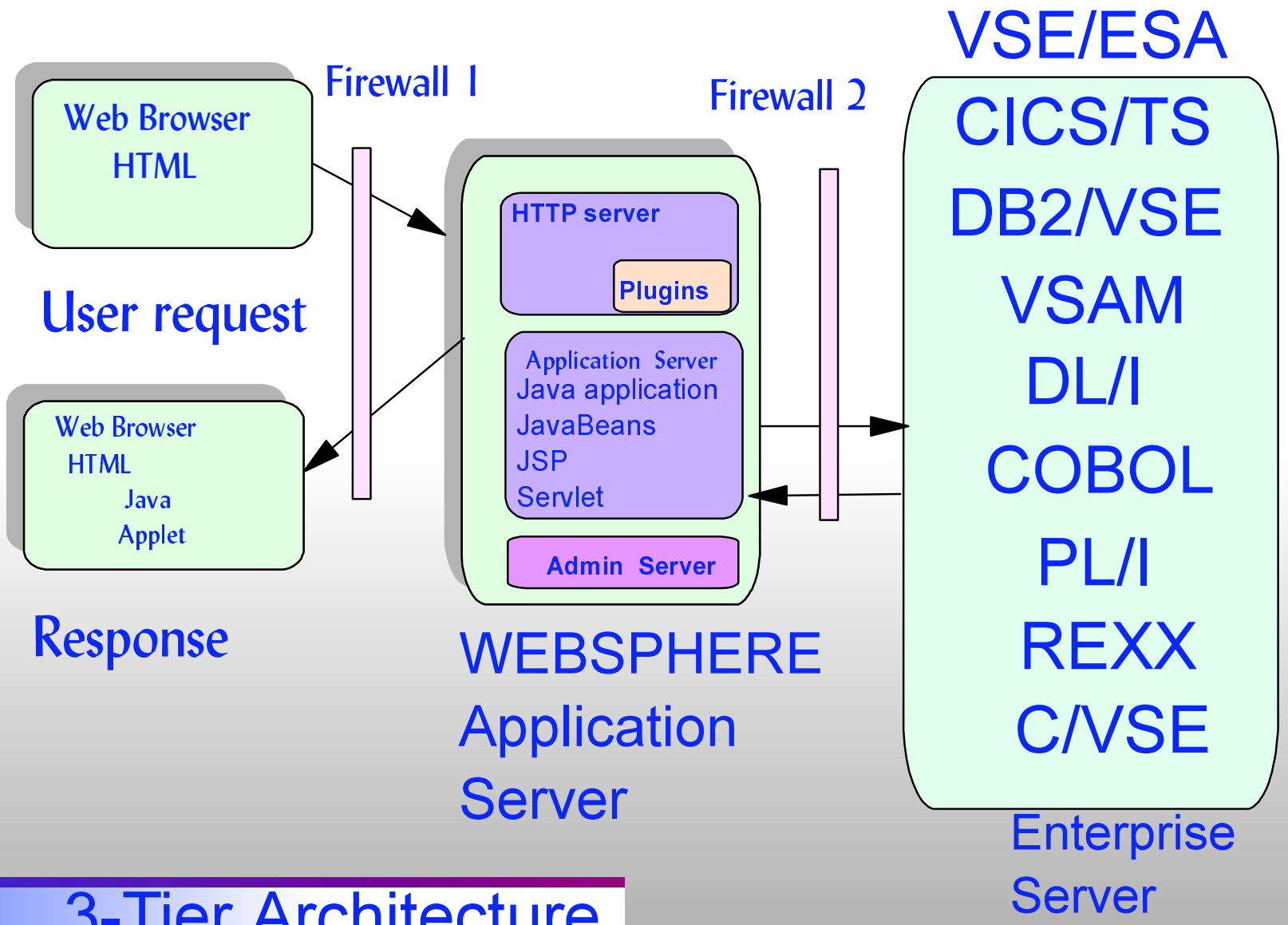
VSE

and YOU!

The following are trademarks of IBM:

IBM	AIX
AS/400	AT
CICS	CT
DB2	DB2 Connect
DB2 Universal Database	Distributed Relational Database Architecture
DRDA	Enterprise Storage Server
Intelligent Miner	MQSeries
Netfinity	Nways
OS/2	OS/390
OS/400	QMF
RS/6000	S/390
SP	System/390
VisualAge	VM/ESA
VSE/ESA	VTAM
WebSphere	Wizard
XT	400
Lotus	1-2-3
Redbooks	Redbooks Logo
Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems Inc in the USA and/or other countries	Other company, product and service names may be trademarks or service marks of others
Windows, Windows NT, Windows 95, Windows 98 are registered trademarks of Microsoft Corporation	

WEBSPHERE - the BIG PICTURE



3-Tier Architecture

VSE and *e-business* together

- VSE/ESA V2.5
  - e-business connectors
    - Easy access to VSE stuff
      - Java based connector
      - DB2 connector
    - Extend your core applications
  - Pick the right platform for the right job
    - VSE/ESA enterprise server
      - Best of breed for data & business
    - WebSphere web application server
      - Cool Tool for the Internet

VSE 2.5 Java based connectors

- "Portfolio" of java-based services
  - Can be used in applets, servlets, JSP's, EJB's or Java applications
    - VSE ***Connector Client***
      - JavaBeans
      - Java samples
      - online documentation
      - Access to VSE file systems
        - VSE/VSAM
        - VSE/POWER
        - VSE librarian
        - VSE/ICCF
        - Operator console

VSE 2.5 Java based connectors

- You will need the Java Development Kit (SDK)
  - Sun recently changed the "name" from
    - JDK (Java Development Kit)

    to
    - SDK (Software Development Kit)
  - Can download this from Sun
    - http://java.sun.com

    Current version is "Java 2 SDK, Standard Edition Version 1.3.0
    - WebSphere may require specific levels of SDK

VSE 2.5 Java based connectors

- You can build Java programs
  - These Java applications can run on the "middle tier" in a 3-tier architecture LIKE MAYBE WebSphere
  - Or the Java applications can run on a different "client"
  - Use VSE JavaBeans to build "connections"
  - You can get "connected" either
    - from PRD1.BASE (which has the file) or at:

www.s390.ibm.com/products/vse/support/vseconn/vsecon.htm

Term time - a few definitions

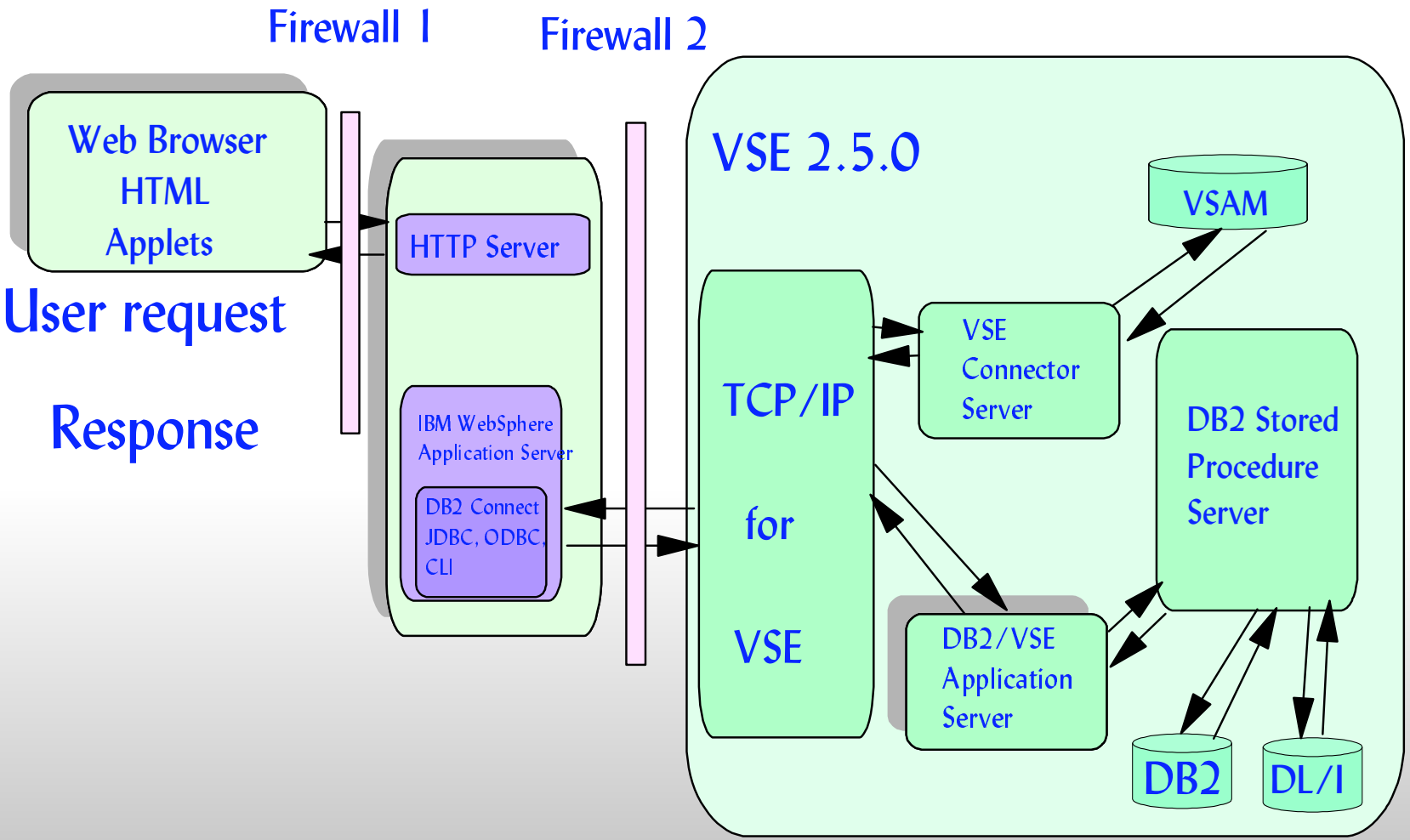
Visit http://java.sun.com/docs/glossary.print.html

Applet	Program written in Java can be embedded inside Web pages and can be executed with a "Java savy" browser on client side
Bean (JavaBean)	Independent, reusable software component. Beans can be combined into an application
Bytecode	Code generated by the Java Compiler. This code is platform independent and must be interpreted by a Java interpreter
Java Archive Files (JAR)	Java file format to compress/combine many files into 1 Can improve transmission time, can "organize" applications
Java Development Kit (JDK) Software Development Kit (SDK)	Sun's software development environment (FREE) SDK is the new term Sun is using
VisualAge Java	Integrated Development Environment for JAVA Visual composition, Editor, Debugger, Compiler, AppletViewer
Java Virtual Machine (JVM)	The Java "interpreter" responsible for interpreting bytecodes
Java Runtime Environment(JRE)	The Java language and the JVM provide means to execute code
Servlet	Similar to an Applet, but located on the SERVER
Java Server Pages	Web pages with embedded JAVA code that allow for "dynamic" content on your web pages

DB2-based connector

- Requires DB2 Server with stored procedure capability
  - Allows access to host data from web browser
    - Uses Java applets to access VSE host data
  - Access to DB2 is via DB2 Connect
  - Access to DI/I is via DB2-based connector and the stored procedure server
  - Access to VSAM is from DB2 stored procedure
    - Uses VSAM Call Level Interface (CLI) function
      - Create "maps" and "views" to describe records

WEBSPHERE and VSE - with connectors



3-Tier Architecture

WebSphere Standard Edition

- WebSphere is IBM framework for e-business with multiple editions for scalability and functionality
- Standard Edition
  - Entry level
  - Support for Java engine
  - Java Server Pages
  - Servlets
  - Connection pooling
  - Multiple JVM support
  - Instant D B (in memory) support for Administration
  - Single node (One WAS host) administration
    - But...you can start here and grow as req'd!

WebSphere Advanced Edition

- Advanced edition has all the stuff from Standard edition PLUS
  - Administration database is now DB/2, Oracle or Sybase
  - Multiple nodes (WAS machines) administration
    - Can "manage" multiple WAS nodes from a single administration server
  - Support for Enterprise Java Beans (EJB's)
    - These are the business logic
  - Workload Management Support (WLM) for servlets and EJB's
  - Failover support
  - Remote Servlet execution support
  - Integrated security

WebSphere Enterprise Edition

- Enterprise edition has all the stuff from Standard edition PLUS
  all the stuff from Advanced Edition plus
  - Full CORBA support
    - Component Broker ORB
    - And if you need CORBA, that is the typical reason most likely to cause you to get Enterprise Edition
    - Support for C++ and Java CORBA business objects
    - Transaction integrity with DB2, Oracle, IMS, CICS and MQSeries
    - Typical to select this version when you already have a HIGH investment in C++

      CORBA - common object request broker

      developed by OMG (Object Management Group)

WebSphere Studio

- Tool to help in your web design
  - Integrated into WebSphere
  - Simplifies your design, development and maintenance of your web sites
  - Standard HTML page design
  - Has tools to help you create
    - JavaBeans
    - Applets
    - Database Queries
    - Servlets
  - and always, management and coordination

Java in your VSE COBOL world

- VisualAge Java is part of WebSphere
  - Or, of course, you can get VisualAge Java independently as a standalone tool
  - VAJAVA is an integrated development tool
  - Seamless integration with WebSphere
  - All-in-one solution for
    - Editing
    - Browsing
    - Execution
    - Debugging
  - Wizards for building
    - EJB's
    - JSP's
    - Java components for WebSphere apps

OK, Let's talk Java and COBOL

- Java
  - is Portable
    - This means write once, run anywhere
      - It also means, write once, TEST EVERYWHERE
  - is definitely Object Oriented
    - Everything is represented as an object
      - Except primitive data types
  - Supports single inheritance
  - Supports polymorphism
  - is just another language to add to your "kit"

COBOL and Java

COBOL	JAVA
Arithmetic operators	Arithmetic Operators
Boolean Operators	Boolean Operators
Built-in FUNCTIONS (INTRINSIC)	Class library
Comments(* in col 7)	Comments (// single line /* multi-line */)
Compute (calculation)	"Assignment"
Condition checking	Boolean True|False
Do Until (Perform with test after)	Do-While
Do While (Perform)	While
Evaluate (case statement)	Switch (case)
If-Then-Else	If-Then-Else
Literals	Literals
Move	"Assignment"
Pass arguments	Pass arguments
Perform varying	For loop
Reserved words	Keywords
Subprograms (Call)	Methods (Invoke)
Table	Array
White Space (blank lines help)	White Space (helps even more in Java)

Java the language

- Objects are just like a cell
  - Self-contained
  - Can be combined to make more complicated things
    - These complicated things can be combined to make an application
  - They inherit from their parents
    - They can be customized to change to be different when necessary
  - Reusability is just one of the keys

So what is this "object-oriented" stuff?

- Object-Oriented is simply looking at our world in terms of things, stuff, aka *objects*
  - Objects have traits and behaviors that identify them
    - How do they look?
    - How do they behave?
    - How do they interact with other objects?
  - An object knows about itself (*data* aka *attributes*)
  - An object knows what it can do (*behavior* aka *methods*)
  - An object provides a way to send/receive messages to communicate with others

More of this "object-oriented" stuff?

- Abstraction
  - Everything that is essential, nothing more
  - When you drive a car do you think about how all the parts work?

    No, you think about starting the car and driving it
    - You send a message via inserting and turning the key
    - You send another message by putting the car in gear (with your foot on the brake)
    - You give the car some gas via your foot on the gas pedal
    - You control the direction via the steering wheel

    Well, you get the idea - these are all messages to your car (object) that may produce a response

More terms

- Class - collection of *data*, stored in named fields, and *code* combined into *methods* to operate on the data
  - Class fields - apply to class, not just 1 instance
  - Class methods - code that applies to the class as a whole, not single instances
  - Instance fields - specific to the specific instance
  - Instance methods - code that operates on an instance (object)

and Still more terms

- Class - This is the template or cookie cutter that is the definition
- Class fields - have the modifier *static*
- Class methods - also have *static modifier*
- Instance fields - variables containing the data that is specific to this instance
- Instance methods - code that does *not* have the *static* modifier (these are the most common, and are simply called methods)
- Instance - one cookie you have "cut" Can have many - access via a reference pointer - this is also an OBJECT!

a VERY simple program

```
public class HelloWorld
{
        public static void main(String args[ ])
        {
        System.out.println("Hello World!");
        }
}
```

or...

a slight variation

```
public class HelloWorld
{
        public static void main(String [ ] args) {
        System.out.println("Hello World!");
        }
}
```

Some of the "rules of the road"

The convention of naming:

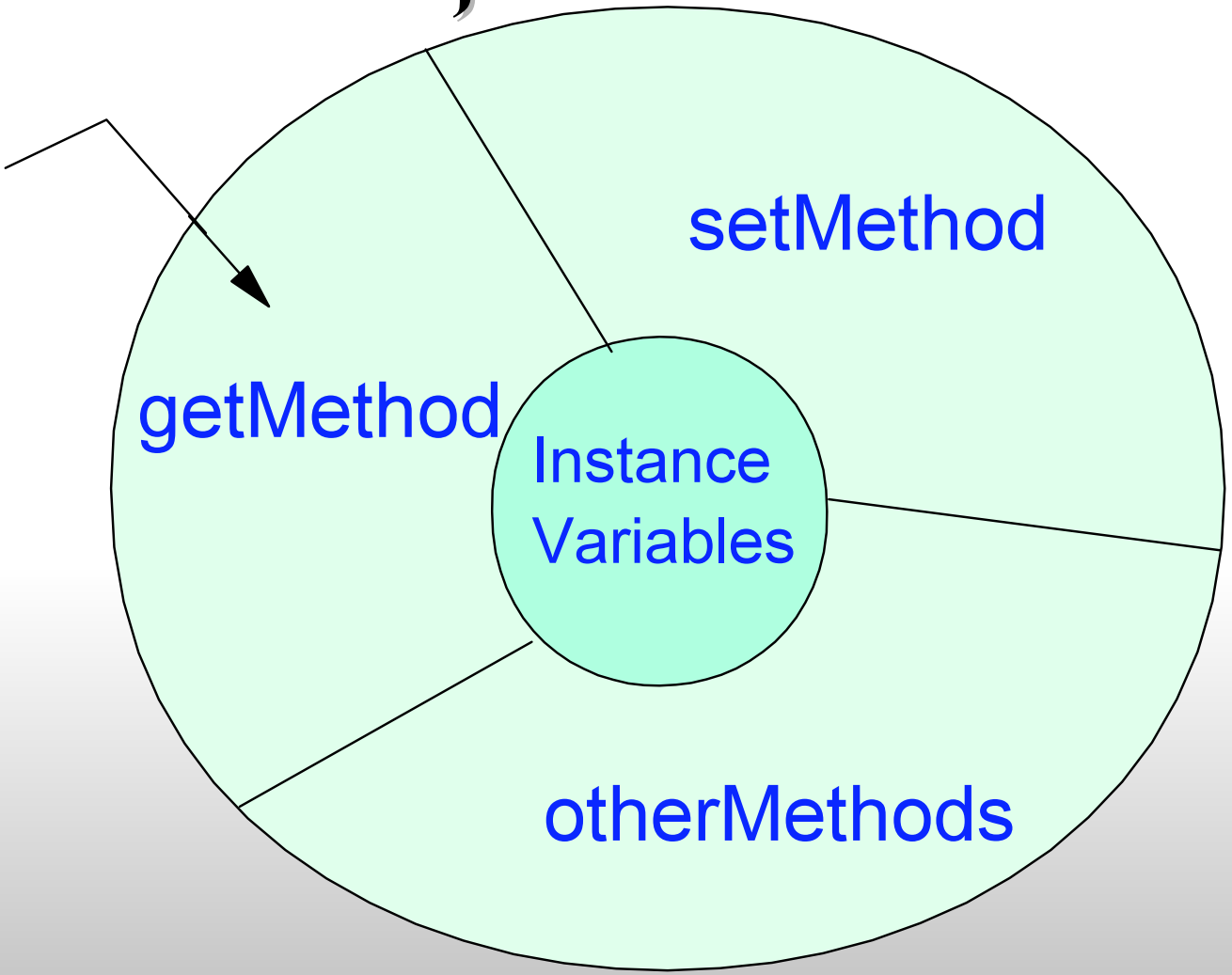
Source files are named the
ClassName.java

Class names are UpperCase for the
first letter of each "part" and especially the
first letter of the ClassName

Method names are lowerCase for the 1st
position, each next part is upperCase

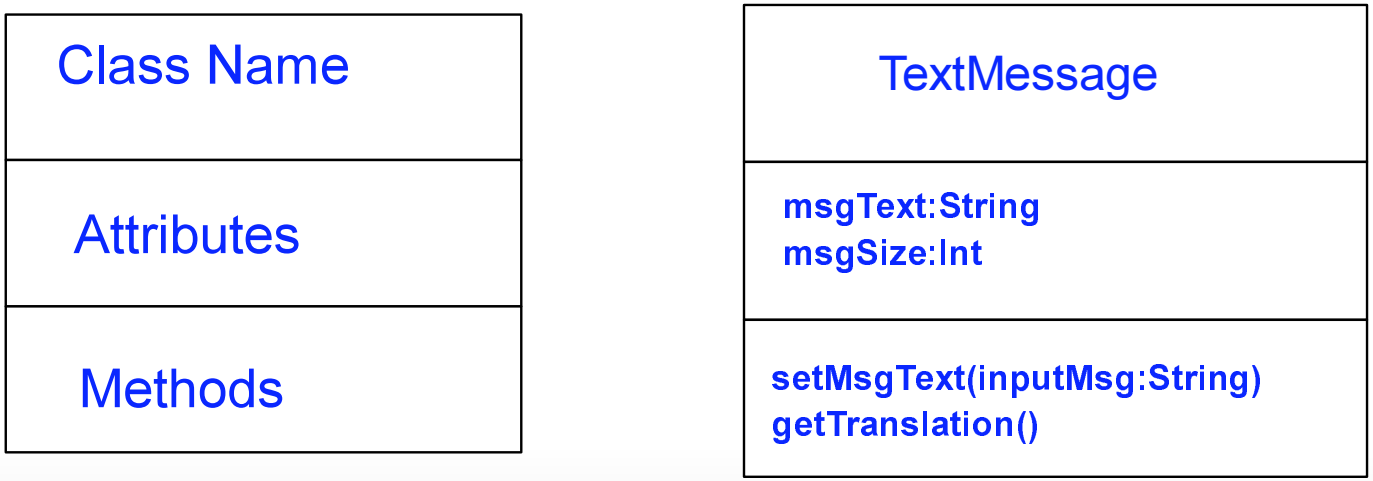
You cannot use a '-' like in COBOL for
names, Java will think this is a subtract

An Object or an Instance



a more recent idea is

Unified Modeling Language (UML)



Can now use class diagrams to show messages and interactions...

Classes typically don't have LOTS of methods

Classes also don't have LOTS of data

but you will have LOTS OF CLASSES!

```
public class HelloWorld
{
        public static void main(String args[])
        {
                String tempMsg;
                System.out.println("Hello World!");

/* Set up an instance of ErrorMsg and call getErrorMsg (no arguments)

                ErrorMsg myErrorMsg = new ErrorMsg ();
                tempMsg = myErrorMsg.getErrorMsg ();
                System.out.println (tempMsg);

                myErrorMsg.setErrorMsg ("Some Text");
                tempMsg = myErrorMsg.getErrorMsg ();
                System.out.println (tempMsg);

                myErrorMsg.setErrorMsg ("Some New Text");
                tempMsg = myErrorMsg.getErrorMsg ();
                System.out.println (tempMsg);

/**     Set up a NEW INSTANCE of an ErrorMsg
         and pass a new text message into the setErrorMsg method
         and then PRINT IT OUT
*/
                ErrorMsg myErrorMsg2 = new ErrorMsg ();
                myErrorMsg2.setErrorMsg ("Some Text for #2");
                tempMsg = myErrorMsg2.getErrorMsg ();
                System.out.println (tempMsg);

//  Call the variation of the 'getErrorMsg' method return UPPER CASE msg

tempMsg = myErrorMsg.getErrorMsg ('U');
                System.out.println (tempMsg);
        }
}
```

the class ErrorMsg (subprogram)

```
public class ErrorMsg {
        public  String msgText = " ";
        public  int    msgSize;

/* Here is the method to simply output the error message
        public void setErrorMsg (String inputMsg) {
                msgText = inputMsg;
        }

/* Here is a method that RETURNS the error message

        public String getErrorMsg () {
                String returnMsg;
                returnMsg = msgText;
                return (returnMsg);
        }

        public String getErrorMsg (char caseFlag) {
                if (caseFlag == 'U')
                        return (getErrorMsg().toUpperCase ());
                else
                        return (getErrorMsg());
        }
}
```

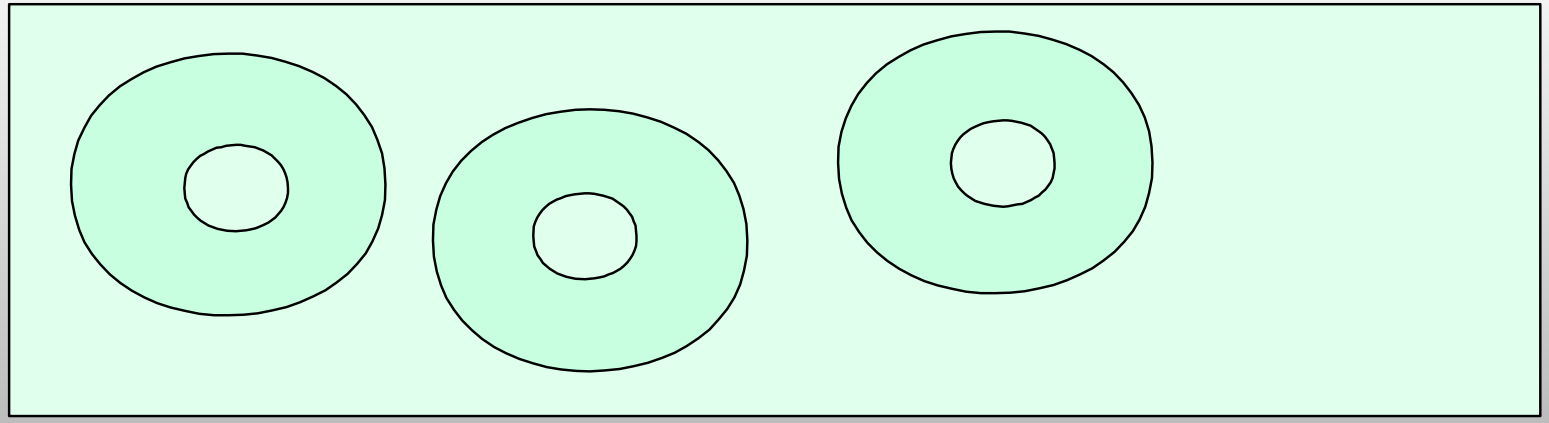
The basics of Java

- The "head" class has a very specific interface:
  public static void main (String args[])
  - This method has a single parameter (args)
    - Must be defined as an array of Strings
    - This string array represents the arguments passed in at runtime
    - The classname can be anything, but it must be public (anyone can get to it) and it must be a static class (only 1 instance at a time)
    - This is just like the program you execute off the JCL, and then call subprograms (additional classes) to "assemble" your app

Static for Class or Variables in your Java Virtual Machine Runtime Environment (JRE)

Static gets created 1 time and has both the main class and any static variables
Then "instances" of the classes that do the specific work are created (instantiated)
These instances may create other instances of other classes
All of these actions are handled via messages
- Traditionally there are getter and setter methods
- HelloWorld is a stand-alone application
- ErrorMsg is a "reusable" class that can be used by any application that needs its "services"

Java supplied LOTS of classes you can USE (don't have to WRITE)



Class stuff

Java is all about CLASSES

- the main() method is the entry point for applications
  - This then invokes the subprograms (methods) which could be in this class or other classes
- It is static so that it can be executed without having to build (construct) an instance of the class
- Class variables are constructed in this same static area, and are available to all instances of this class
- Instance variables can be initialized when the instance is created (instantiated) or via assignment (=) in a method

Java explored

- Java Program is collection of interacting classes
  - Class with MAIN always is the starter
  - Other classes invoked to perform specific tasks
  - Class is a template or blueprint
    - Objects are created (instantiated)
      - This creates an instance of a class
- Java development environment supplies classes
  - Use them as is, or adapt them (modify)
    - Subclassing extends the functionality
      - REUSES what is there, adds what is NOT there

Java explored

- Package
  - Group of "related" classes
  - Similar to organizing members in a library
    - In VSE we organize similar stuff into a library
    - Then we use a LIBDEF to find the individual parts
  - Java has a similar organization
    - Classes in the same package can "call" each other
    - If a class is in a different package
      - Now you import the required packages
        - This is very like LIBDEF to multiple libraries for access

Let's talk COBOL and JAVA

Java uses the same arith operators as COBOL the calculation is via an "assignment"

```
COBOL
   Compute Balance = Balance - Check-Amt
   Compute Int-Amt = Loan-Amt * Int-Rate
```

```
JAVA
   balance = balance - checkAmt
   intAmt = loanAmt * intRate
```

Let's talk COBOL/JAVA

Java code	Shortcut	COBOL code
i = i + 1;	i++1;	Compute I = I + 1
i = i - 1;	i--1;	Compute I = I - 1
i = j + 1;	i = ++j;	Compute J = J + 1
i = j - 1;	i = --j;	Compute J = J -1
i = j;		MOVE J to I
%	Modulus	Function MOD
I++	Post-increment	no real COBOL match here
++i	Pre-increment	no real COBOL match

Let's talk COBOL/JAVA

Java code	COBOL code
<	< or Less than
>	> or Greater Than
<=	<= Less than or equal to
>=	>= Greater than or equal to
==	= Equal to
!=	Not equal
||	or
&&	and
!	NOT

Note that in Java an = sign is the assignment operator not an equals to

Let's talk more COBOL/JAVA

Java "logical expression" vs COBOL "condition"

Java code

```
if (minimumBalance < 500) serviceCharge = 5.00F;
   else
    serviceCharge = 0.00F;
```

COBOL code

```
IF Minimum-Balance < 500
   MOVE 5.00 to Service-Charge
ELSE
   MOVE 0.00 to Service-Charge
END-IF
```

Let's talk more COBOL/JAVA

Java switch vs COBOL EVALUATE

Java code

```
switch (variable)
{
   case value-1: statement-1;
      break;
   case value-2: statement-2;
      break;
   default; statement-3
}
```

IMPORTANT:
Java switch must be
variable of type
integer or character
must use break to exit
must use colon, semi-colon
and always curly braces!

COBOL code

```
EVALUATE variable
 WHEN value-1
   statement-1
   statement-2
 WHEN value-x
   statement-a
   statement-b
  WHEN OTHER
    statement-z
END-EVALUATE
```

```
EVALUATE variable1 also variable2 also TRUE
 WHEN      value-1   also  value-2   also  WS-AMT < 500
   statement-1
   statement-2
 WHEN      value-x   also   any       also   88-level
   statement-a
   statement-b
  WHEN OTHER
    statement-z
END-EVALUATE
```

Let's talk more COBOL/JAVA

Java vs COBOL looping

- COBOL
  - PERFORM UNTIL
  - Inline PERFORM
  - PERFORM...VARYING..UNTIL
  - PERFORM..VARYING..UNTIL..AFTER
- Java
  - while
  - for
  - do
  - break
  - continue (which is totally different - this terminates the current loop)

Let's talk more COBOL/JAVA

Java vs COBOL looping

Java code

```
while (boolean_expression)
{
statements to be done
while the boolean expression
is true
}
```

When the boolean expression is *false* the loop terminates

```
int aNum = 1;
while (aNum <= 5)
{
  System.out.println (aNum);
  aNum = aNum + 1;
}
```

COBOL code

```
PERFORM UNTIL condition
    statements
    statements
      END-PERFORM
```

When the condition is *true* the COBOL loop terminates

```
MOVE 1 TO A-NUMBER
PERFORM UNTIL A-NUMBER > 5
  DISPLAY A-NUMBER
  ADD +1 TO A-NUMBER
END-PERFORM
```

Let's talk more COBOL/JAVA

Java vs COBOL looping

Java code

```
for (initialize_statement; boolean_expression; increment_statement)
{
statements to be done
until the boolean expression
is false
}
```

```
for (int aNum = 1; aNum <= 5; aNum = aNum + 1)
{
  System.out.println (aNum);
}
```

The for expression must be in parentheses
Always semicolon, curly braces!

COBOL code

```
PERFORM VARYING variable
  FROM init-value BY increment
   UNTIL condition
     statements
     statements
END-PERFORM
```

When the condition is *true*
the COBOL loop terminates

```
MOVE 1 TO A-NUMBER
PERFORM UNTIL A- NUMBER  >  5
  DISPLAY A-NUMBER
  ADD +1 TO A-NUMBER
END-PERFORM
```

Let's talk more COBOL/JAVA

Java and COBOL and TABLES or ARRAYS

Java code

```
int numberOfThings [ ];

/* no memory is set up yet, no "size" - simply  tells JAVA we intend to
   create an array and we will finish it....
*/

numberOfThings = new int [12]

//   or combine the above 2 statements and do it all at once

int numberOfThings[ ] =  new numberOfThings[12]

/* in JAVA and ARRAY can hold any data type (the occurrences just
   have to all be the same type.
   An array is an OBJECT, so in order to use it you must
   instantiate it - that means create an OBJECT with NEW!
   Also note that in JAVA the 1st element in an ARRAY is element [0]
*/
```

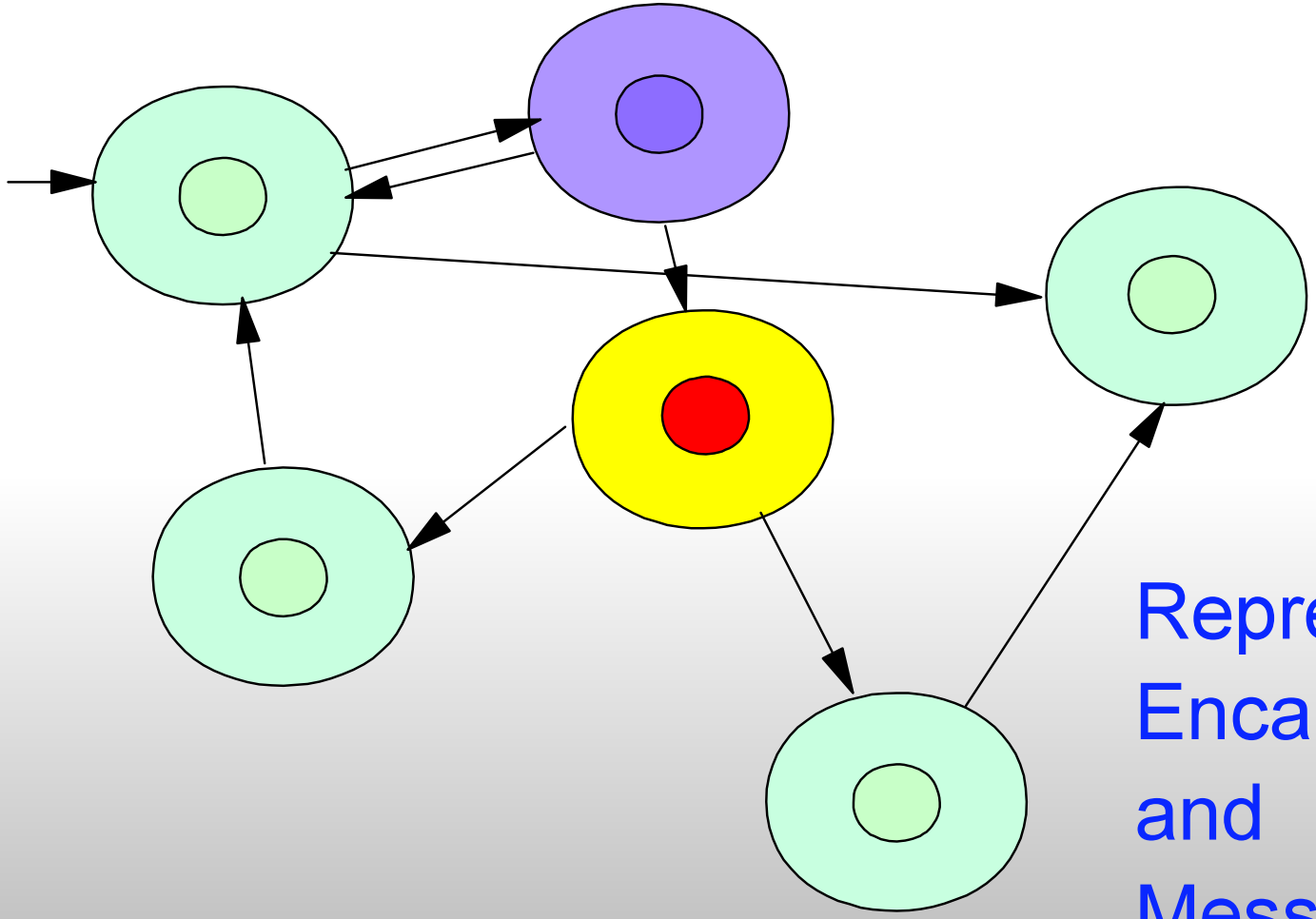
Java and COBOL and TABLES or ARRAYS

Java code

```
// The following statement is essentially an inline initialization of an array

int numberOfThings [ ] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

// or you can initialize like this:

numberOfThings [0] = 1;
numberOfThings [1] = 2;

// and on and on
// notice and remember that the 1st element is 0 not 1!

String monthNames [ ] = {"Jan", "Feb"," Mar", "Apr"....};

int monthNum;                    // use month number as index
monthNum = 0;                    // index is 0 for January, etc
while (monthNum < 12 (NumberOfThings[monthNum]
{
 System.out.println(numberOfThings[monthNum] + "    " +
   monthNames [monthNum];
monthNum = monthNum + 1;
}
```

One view of a Java Application

Or - Objects communicating via messages



Representing Encapsulation and Messaging

WebSphere, Java, and VSE

WebSphere provides a great set of tools

Java is just one

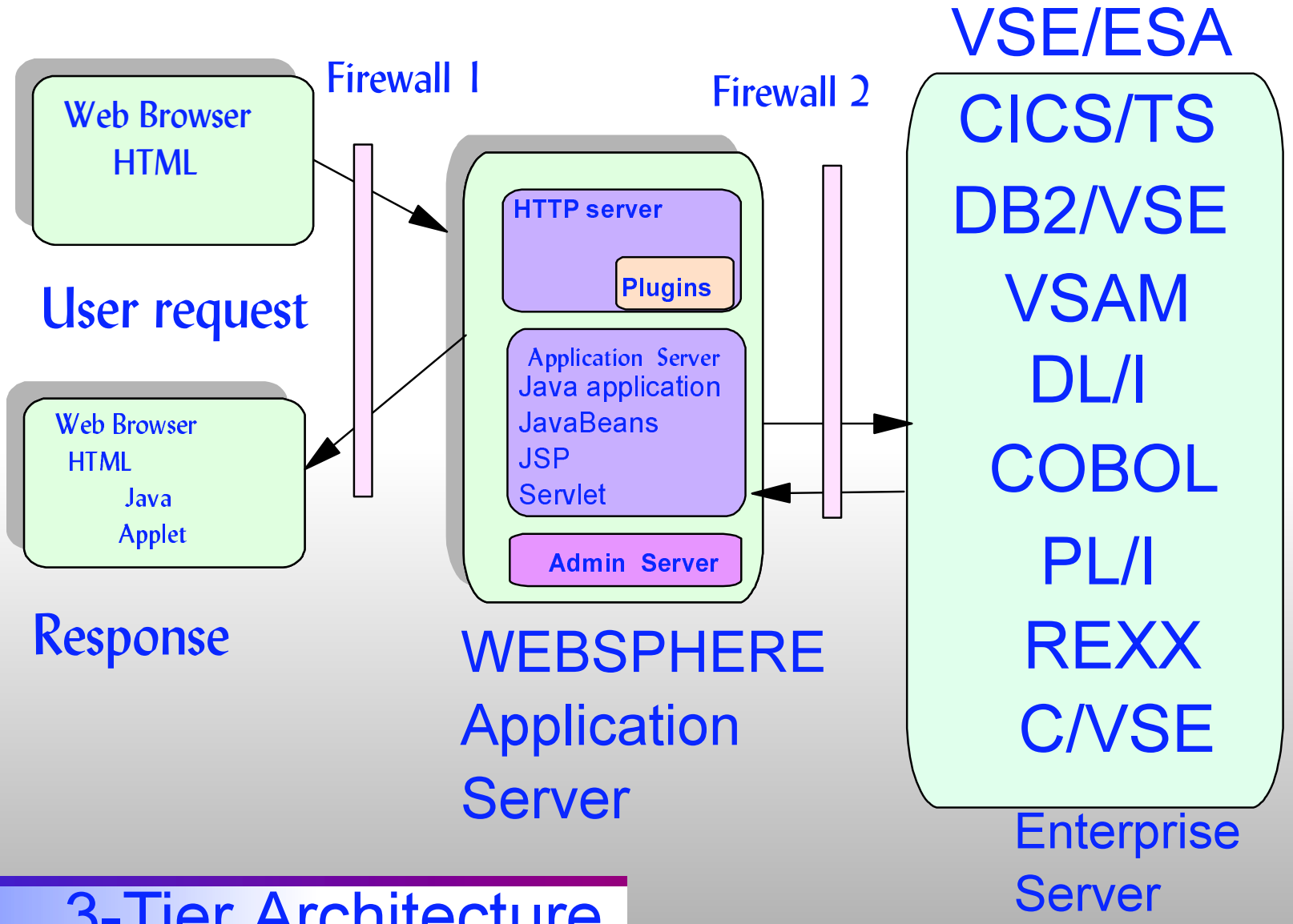
Regardless of the tools you pick

ANALYSIS and DESIGN STILL COUNT

and that is something you already know how to do!

WEBSPHERE, Java, VSE and You

the BIG PICTURE should now be a clearer picture!



3-Tier Architecture

Hopefully now you have more

- Understanding of what WebSphere is/does
- Confidence that JAVA is just another language
  - True, the syntax is different, and the style is a bit different, but the stuff is still there
- Comfort that VisualAge for Java could help you extend and expand from VSE to the WEB
- Knowledge about how these products might "fit" into connecting your VSE system to the WEB