INFO213: Lecture 3

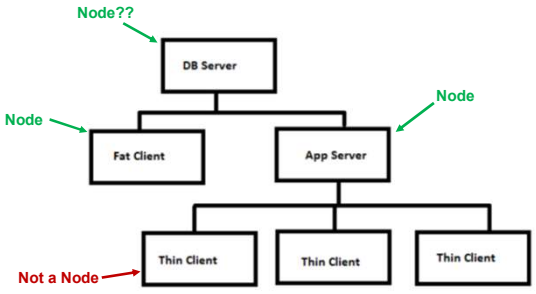
JADE Fundamentals

JADE Nodes, JADE Caching, JADE Syntax, JADE Access Control, JADE Painter

First on the agenda:

- Nodes, Processes and Caches
  - Complicated!
  - Don't worry, we will keep touching on it...
- Persistent vs Transient objects
- Cache coherency

A JADE System: Find the nodes!



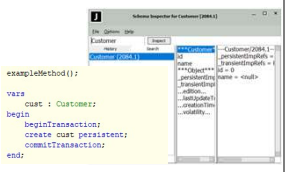
What's a Node

*A node is a component of a JADE system where **application code** is executed and where **objects are processed**.*

- A node has two caches:
  - A Persistent Cache – Stores *copies* of Persistent Objects in memory
  - A Transient Cache – Stores Transient Objects in memory
- A Node can also run any number of JADE Processes, which are what the application code runs on.

Persistent vs Transient Objects

Persistent



```
exampleMethod();

vars
    cust : Customer;
begin
    beginTransaction;
    create cust persistent;
    commitTransaction;
end;
```

- Persistent Objects are saved to the database
- As such, they are **long-lived**.
- They can be created and modified only inside **beginTransaction; commitTransaction; blocks**

Transient

```
exampleMethod();

vars
    cust : Customer;
begin
    create cust transient;
    // Do something interesting with the customer...

epilog
    delete cust;
end;
```

- Transient Objects are stored in memory
- As such, they are **short-lived**.
- You do not need any transaction for them, but you SHOULD delete them when you're done with them.

Server Execution

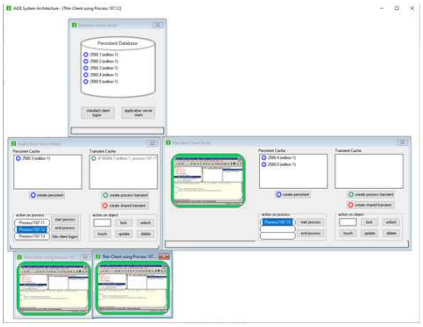
serverExecution Option

The **serverExecution** method option indicates that the method and all methods subsequently called by this method are to be executed on the server node unless they are **clientExecution** methods, in which case they are executed on the node of the client calling the method. This method option provides performance benefits when a method accesses a large number of objects in multiuser mode, as the methods are executed on the node in which the objects reside instead of the required objects having to be passed to the client node for execution.

The method in the following example iterates through 5,000 entries in the **Number** collection and executes using the **serverExecution** method option.

```
executeOnServer() updating, serverExecution;
vars
    num : Number;
begin
    // This method should take less time to execute than the same method
    // executing using the clientExecution option.  When this method was
    // tested on a presentation client running in JADE thin client mode,
    // it took only 1.602 seconds to execute.
    beginTransaction;
    foreach num in numbers do
        num.key := 0;
    endforeach;
    commitTransaction;
end;
```


Cache Coherency - Demo



JADE Syntax

- Hopefully, you already have already practiced much of this in tutorials!
- We're going to go through this speedily – can come back these slides for notes if needed!

Flow control – If, Foreach, While

- **If** statements control whether code is executed based on a condition (true/false)
- **Foreach** loop performs one block of code multiple times – once for each element in a range or collection
- **While** loop performs one block of code multiple times – as many times as needed to make a condition false
  - Be careful! It's very easy to get stuck in an infinite loop when using these!

Flow Control - Break and Continue

- Use these while inside a loop (foreach or while)
- **Break** exits out of the loop immediately
- **Continue** merely skips the rest of that iteration of the loop, beginning again at the top of the loop.

Method Parameters

```
testCensor();

vars
    banned : StringArray;
begin
    create banned transient;
    banned.add("foo");
    banned.add("bar");

    write censorMessage("The quick brown Foo jumped over the lazy bar", banned);

epilog
    delete banned;
end;
```

```
censorMessage(message : String; bannedWords : StringArray) : String;

vars
    bannedWord : String;
    censoredStr : String;
begin
    censoredStr := message;

    foreach bannedWord in bannedWords do
        censoredStr := censoredStr.replace__(bannedWord, "[REDACTED]", true);
    endforeach;

    return censoredStr;
end;
```


Method Parameters – Usage Options

```
called(pString: String constant; pCust: Customer constant);

begin
    pString := "Hello World";                   // NOT allowed
    pString.replaceChar("a", "b");              // NOT allowed
    pCust := Customer.firstInstance();          // NOT allowed
    pCust.address := "Smallville";              // NOT allowed
end;
```

```
called(pString: String input; pCust: Customer input);

begin
    pString := "Hello World";                   // NOT allowed
    pString.replaceChar("a", "b");              // Allowed
    pCust := Customer.firstInstance();          // NOT allowed
    pCust.address := "Smallville";              // Allowed
end;
```

```
called(pString: String output; pCust: Customer output);

begin
    pString := "Hello World";                   // Allowed
    pString.replaceChar("a", "b");              // Allowed
    pCust := Customer.firstInstance();          // Allowed
    pCust.address := "Smallville";              // Allowed
end;
```

There is also an io option, which is like output but not nulled on entry…

String manipulation : Indexing

```
stringManip();

vars
    loremIpsum : String;
begin
    loremIpsum := "Lorem ipsum dolor sit amet, consectetur adipiscing elit." & CrLf
                & "Duis tellus neque, hendrerit eget lacinia quis, tempus nec odio.";

    write loremIpsum[3]; // "r"
    write loremIpsum[3:7]; // "rem ips"
    write loremIpsum[10]:end]; // " quis, tempus nec odio."
    loremIpsum[1:5] := "merol"; // Changes the first word to "Merol"
    loremIpsum[1] := "Z"; // Changes the first character to "Z"
end;
```


String manipulation : Common methods

```
stringManip();

vars
    loremIpsum : String;
begin
    loremIpsum := "Lorem ipsum dolor sit amet, consectetur adipiscing elit." & CrLf
                & "Duis tellus neque, hendrerit eget lacinia quis, tempus nec odio.";

    write loremIpsum.pos("ipsum", 1); // 7
    write loremIpsum.pos("ipsum", 8); // 0 (not found)
    write loremIpsum.reversePos("o"); // 121 (looks from back)
    write loremIpsum.reverse(); // one guess what this does
    write loremIpsum.replace__("o", "0", true); // replaces all o with 0
    write loremIpsum.trimBlanks(); // strips whitespace from begin and end of string
end;
```

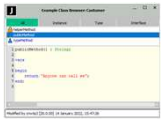
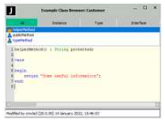
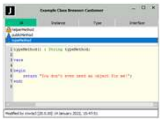

Encapsulation (Access Control)

- Encapsulation
  - What is a capsule?
  - Why do we put things in capsules?

JADE Property Access

Public	Protected	Read Only
		
Methods of this class read? ✓	Methods of this class read? ✓	Methods of this class read? ✓
Methods of this class update? ✓	Methods of this class update? ✓	Methods of this class update? ✓
Methods of other classes read? ✓	Methods of other classes read? ✗	Methods of other classes read? ✓
Methods of other classes update? ✓	Methods of other classes update? ✗	Methods of other classes update? ✗

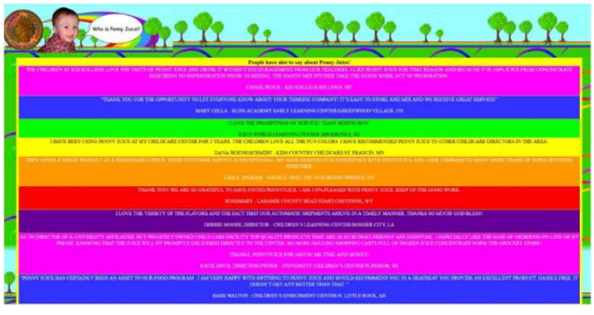
JADE Method Access

Public	Protected	Type Method
		
Part of the class's public interface. Can be called from any method with an object of this class.	Not part of the class's public interface. Can only be called from methods of the same object. Useful for "helper" methods, which split up one longer method into multiple smaller or reuse a common bit of code.	Doesn't require an object of the class to use (can be called directly from the class using ClassName@methodName syntax Used for utility methods. Can't use any properties or non-type-methods of the class.

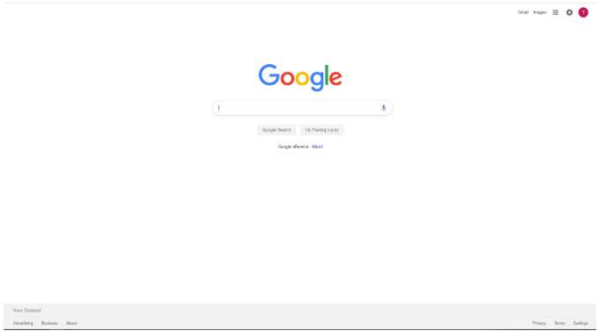
GUIs and the JADE Painter

- GUI = Graphical User Interface
- The JADE Painter allows you to design GUIs.
- GUIs aren't the only sort of interface
  - APIs (Application Programming Interface)
  - CLIs (Command-Line Interface)
- GUIs can be good or bad … How do we measure quality of a GUI?

Is this a good GUI? What's wrong with it?



How about this one?



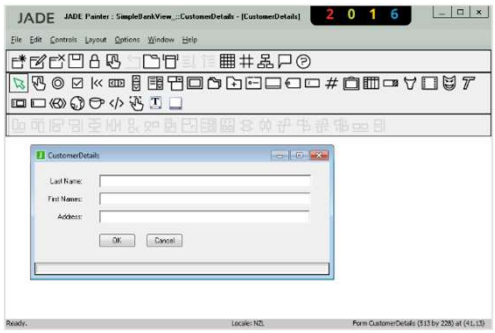
How do we make our UI *usable*?

1. Learnability - ???
2. Memorability - ???
3. Efficiency - ???
4. Error Recovery - ???
5. Satisfaction - ???

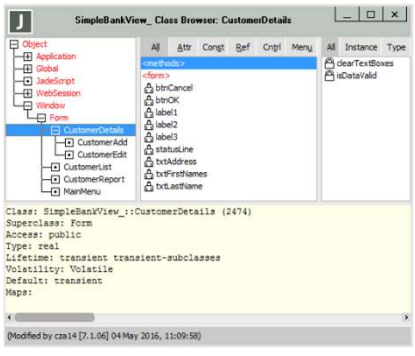
How do we make our UI *usable*?

1. Learnability - How easy is it for newbies?
2. Memorability - What about 2nd time?
3. Efficiency - How fast can I go?
4. Error Recovery - What if I mess up?
5. Satisfaction - Is it pleasant to use?

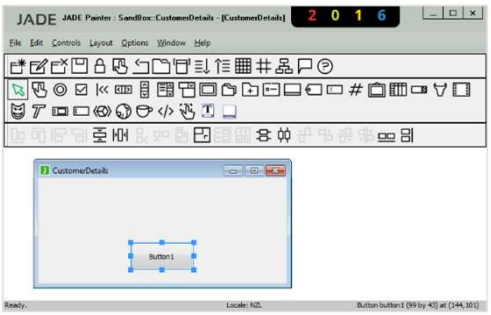
GUI Design+Development in JADE



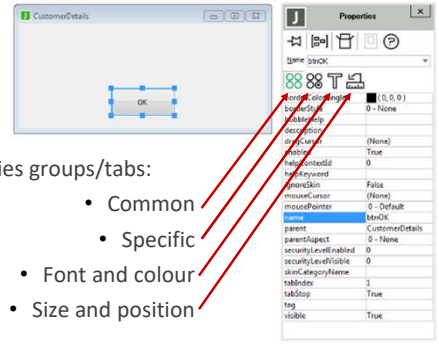
Seamlessly integrates with JADE code



Point and Click Interface Component Placement



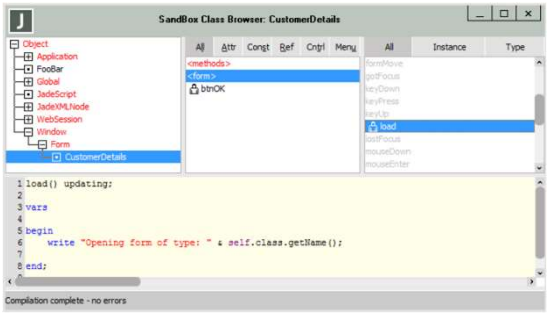
Properties Dialog (E.g., for a Button)



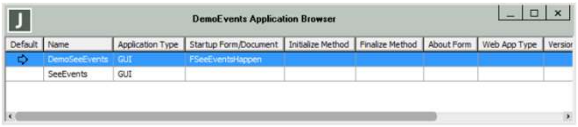
Properties groups/tabs:

- Common
- Specific
- Font and colour
- Size and position

Inherited Method Overriding

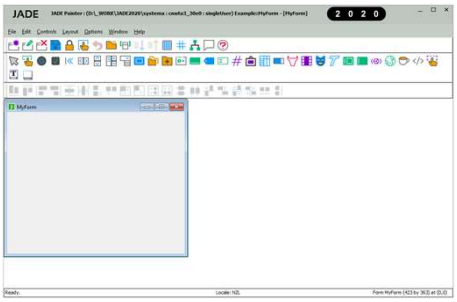


JADE IDE: Applications Browser



- Application Browser brings together all applications defined in a schema
- GUI and non-GUI applications options can be defined in the same schema

DEMO: Creating a JADE Form



Action Points, Readings for Next Week

- Make a start on the assignment!
  - Consider what extra information you might need…
  - Create the classes/references as per the class diagram
  - Start thinking about what the user interfaces will look like
  - Make a use case diagram?