

# M500 Revision Weekend

## - Saturday Afternoon

---

- Building Blocks
- Learning from Mistakes
- Collection Classes

# Building Blocks

---

- Blocks introduced in Chapter 16:

```
[Dialog warn: 'Anyone there?']
```

- An instance of the class **BlockClosure**.

- Message required to cause evaluation:

```
[Dialog warn: 'Anyone there?'] value
```

or:

```
true ifTrue: [Dialog warn: 'It is true!']
```

# Building Blocks

---

- Like other objects, Blocks can be assigned to variables:

```
| myBlock |  
myBlock := [Dialog warn: 'Anyone there?'].  
myBlock value
```

- Messages may take a Block as an argument:

```
( boolean expression ) ifTrue: [ block ]
```

and even as the receiver:

```
[ boolean expression block ] whileFalse: [ block ]
```

# Building Blocks

- Blocks may be thought of as “unnamed methods” and used to construct code that is easier-to-read:

```
firstGuess
  "Determine if item is an animal, mineral or vegetable.
  Categorize and answer the receiver."

  | prompt guess |

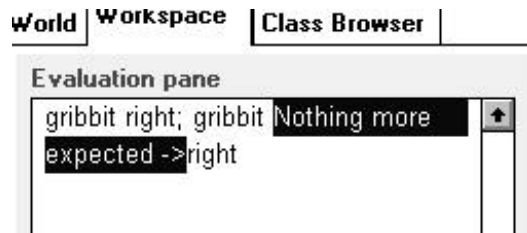
  prompt := [ guess := Dialog
              request: 'animal, mineral or vegetable?'
              initialAnswer: ''
              onCancel: [ Dialog warn: 'We'll assume an animal'.
                          'animal' ]
              ].

  [ (guess = 'animal') | (guess = 'mineral') | (guess = 'vegetable') ]
  whileFalse: [ prompt value ].

  self categorize: guess
```

# Learning from Mistakes

- Programming errors may be classified as:-



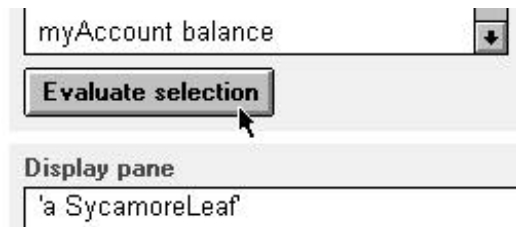
Syntax errors - Error in format

`gribbit right; gribbit right`



Semantic errors - Error in meaning

`gribbit overLimit`



Logical errors - Error in methodology

`myAccount balance`

# Learning from Mistakes

---

- An error that occurs during runtime raises an exception.
- Smalltalk programs are compiled into byte-code which runs on a Virtual Machine. This makes:-
  - final code more portable,
  - development and running programs safer,
  - performance slower than native code.

# Collection Classes

---

- Provide a rich set of dynamic **data types**
- Common protocol (via **polymorphic** messages)
- Classes encountered include:
  - String,
  - Symbol,
  - Array,
  - Set,
  - Bag,
  - Dictionary,
  - OrderedCollection (used to implement a **stack**),
  - SortedCollection.

# Collection Class Methods

Class	Accessing	Adding	Removing	Testing	Enumerating
Array String Symbol	size at: at:ifAbsent: at:put:	N/A	N/A	includes: isEmpty	do: keysAndValuesDo:
Set	size	add:	remove: remove:ifAbsent:	includes: isEmpty	do: detect:ifNone:
Dictionary	size at: at:ifAbsent: at:put:	add:	removeKey: removeKey:ifAbsent:	includes: includesKey: isEmpty	do: keysAndValuesDo: detect:ifNone:
OrderedCollection	size at: at:put: first last	add: addFirst: add:after: add:before: add:beforeIndex:	remove: remove:ifAbsent: removeFirst removeLast	includes: isEmpty	do: detect:ifNone:
SortedCollection	size at: first last	add:	remove: remove:ifAbsent: removeFirst removeLast	includes: isEmpty	do: detect:ifNone: