CLIPS Tutorial 5
THE DEFMODULE CONSTRUCT

- CLIPS uses the defmodule construct to partition a knowledge base by defining the various modules.
  
  (defmodule <module-name> [<comment>])

- By default CLIPS defines a module called the MAIN module.
  
  CLIPS> (clear)
  CLIPS> (deftemplate sensor (slot name))
  CLIPS> (ppdeftemplate sensor)
  (deftemplate MAIN :: sensor
  (slot name))
  CLIPS>

- The :: symbol in the name MAIN::sensor is called the module separator.

- By default, CLIPS places newly defined constructs in the current module.
THE DEFMODULE CONSTRUCT

Whenever a new module is defined, CLIPS makes it the current module.

```
CLIPS> (defmodule DETECTION)
CLIPS> (defmodule ISOLATION)
CLIPS> (defmodule RECOVERY)
CLIPS> (defrule example1 =>)
CLIPS> (ppdefrule example1)
    (defrule RECOVERY: :example1
        =>)
CLIPS>
```

The module in which a construct is placed can be specified in the construct's name.

```
CLIPS> (defrule ISOLATION::example2 =>)
CLIPS> (ppdefrule example2)
    (defrule ISOLATION: :example2
        =>)
CLIPS>
```
THE DEFMODULE CONSTRUCT

- The current module is changed when the module name is specified in a construct’s name.
- The current module can be determined with the get-current-module function.
- The function set-current-module is used to change the current module.

```
CLIPS> (get-current-module)
ISOLATION
CLIPS> (set-current-module DETECTION)
ISOLATION
CLIPS>
```
Facts themselves can also be partitioned.

```clips
(deftemplate DETECTION::fault
  (slot component))
(assert (fault (component A)))

(facts)
f-0 (fault (component A))
For a total of 1 fact.

(deftemplate ISOLATION::Possible-failure (slot component))
(assert (possible-failure (component B)))

(facts)
f-1 (possible-failure (component B))
For a total of 1 fact.

(set-current-module DETECTION)
```

ISOLATION

```clips
(facts)
f-0 (fault (component A))
For a total of 1 fact.
```
The `facts` command, like the `list-defrules` and similar commands, can accept a module name as an optional argument.

```
(facts [<module-name>] [<start> [<end> [<maximum>]]])
```

```
CLIPS> (facts DETECTION)
f-0 (fault (component A))
For a total of 1 fact.
CLIPS> (facts ISOLATION)
f-1 (possible-failure (component B))
For a total of 1 fact.
CLIPS> (facts RECOVERY)
CLIPS> (facts *)
f-0 (fault (component A))
f-1 (possible-failure (component B))
For a total of 2 facts.
CLIPS>
```
IMPORTING AND EXPORTING FACTS

- Unlike defrule and deffacts constructs, deftemplate constructs (and all facts using that deftemplate) can be shared with other modules.
- A fact is “owned” by the module in which its deftemplate is contained, but the owning module can export the deftemplate associated with the fact.
- In order to use a deftemplate defined in another module, a module must also import the deftemplate definition.
IMPORTING AND EXPORTING FACTS

- The export attribute must use one of the following formats:
  - `(export ?ALL)`
  - `(export ?NONE)`
  - `(export deftemplate ?ALL)`
  - `(export deftemplate ?NONE)`
  - `(export deftemplate ?<deftemplate-name>+)`

- The first format will export all exportable constructs from a module.
- The second format indicates that no constructs are exported and is the default for a defmodule.
- The third format indicates that all deftemplates in a module are exported.
- The fourth format indicates that no deftemplate constructs are exported.
- The fifth format gives a specific list of deftemplates exported by a module.
IMPORTING AND EXPORTING FACTS

The import attribute also has five possible formats:

(import <module-name> ?ALL)
(import <module-name> ?NONE)
(import <module-name> deftemplate ?ALL)
(import <module-name> deftemplate ?NONE)
(import <module-name> deftemplate <deftemplate-name>+)
To illustrate importing and exporting facts, let's assume that the RECOVERY module wants to import the `fault` deftemplate from the DETECTION module and the `possible-failure` deftemplate from the ISOLATION module.

(defmodule DETECTION
    (export deftemplate fault))
(deftemplate DETECTION: : fault (slot component))

(defmodule ISOLATION
    (export deftemplate possible-failure))
(deftemplate ISOLATION: : possible-failure (slot component))

(defmodule RECOVERY
    (import DETECTION deftemplate fault)
    (import ISOLATION deftemplate possible-failure))
IMPORTING AND EXPORTING FACTS

CLIPS>
(deffacts DETECTION::start
  (fault (component A)))
CLIPS>
(deffacts ISOLATION::start
  (possible-failure (component B)))
CLIPS>
(deffacts RECOVERY::start
  (fault (component C))
  (possible-failure (component D)))
CLIPS> (reset)
CLIPS> (facts DETECTION)
f-0 (fault (component A))
f-2 (fault (component C))
For a total of 2 facts.
CLIPS> (facts ISOLATION)
f-1 (possible-failure (component B))
f-3 (possible-failure (component D))
For a total of 2 facts.
CLIPS> (facts RECOVERY)
f-0 (fault (component A))
f-1 (possible-failure (component B))
f-2 (fault (component C))
f-3 (possible-failure (component D))
For a total of 4 facts.
CLIPS>