Struct and Comparable

Struct

The quickest way to start a class

Making StructClass

Named

```
Struct.new("Card",:rank,:suit)
```

```
1.9.3p125 :003 > Struct.new("Card",:rank,:
suit)
 => Struct::Card
1.9.3p125 :004 > Struct.constants
 => [:Tms, :Card]
```

What's Struct::Tms?

Struct for storing process System and User times and child processes System and User Times.

```
1.9.3p125 :007 > Struct::Tms.new.inspect
 => "#<struct Struct::Tms utime=nil, stime=nil, cutime=nil, cstime=nil>"
```

Making StructClass

Anonymous, doesn't add to Struct.constants

```
Card = Struct.new(:rank,:suit)
```

Adding on to a StructClass

```
class Card < Struct.new(:rank,:suit)
  def to_s
    "#{rank}#{suit}"
  end
  alias old_inspect inspect
  alias inspect to_s
end
```

Struct implements Enumerable

```
class Card < Struct.new(:rank,:suit)
  def to_s
    collect(&:to_s).join
  end
  alias old_inspect inspect
  alias inspect to_s
end
```

Struct implements inspect

```
1.9.3p125 :012 > Card.new(:A,:H).old_inspect
 => "#<struct Card rank=:A, suit=:H>"
```

Constructing instances

```
Card.new(:A,:H) == Card[:A,:H]
 => true
```

Struct accessors

```
1.9.3p125 :019 >   c= Card.new(:A,:H)
 => #<struct Card rank=:A, suit=:H>
1.9.3p125 :020 > c.rank
 => :A
1.9.3p125 :021 > c.suit
 => :H
1.9.3p125 :022 > c[:rank] # or c["rank"]
 => :A
1.9.3p125 :023 > c[0]
 => :A
```

Struct setters

```
1.9.3p125 :026 > c["rank"] = 5
 => 5
1.9.3p125 :027 > c.suit = :S
 => :S
1.9.3p125 :028 > c[1] = :D
 => :D
1.9.3p125 :029 > c
 => #<struct Card rank=5, suit=:D>
```

Struct miscellany

each, each_pair, select,
hash
length/size
members
to_a/values
values_at (ranges or indices)

Comparable

Makes Ruby object ordering natural

Comparable contract

Implement <=> (spaceship operator)

and you get
>, >=, <, <=, ==, between?(min,max)

also used by default in .sort

Making spaceships

<=> returns -1, 0, 1

-1 self less than other object
0 self equal to other object
1 self greater than other object

Partially Ordered

David and I think partially ordered objects shouldn't really use comparable's == implementation.

Comparable example

```ruby
class Card < Struct.new(:rank,:suit)
  include Comparable
  Ordering = (2..10).to_a + [:J, :Q, :K, :A]
  def <=>(other_card)
    Ordering.index(rank) <=> Ordering.index(other_card.rank)
  end
end
```

High card poker hand

Example of the high card poker hand kata done by Alex using Comparable

Questions?