RENO COLLECTIVE

FULL SPEED

N V

AHEAD

ACADEMY

Q&A

Ternary Operator

Three Part Conditional

```
def unlock_door
  puts "door unlocked"
end

def display_error
 puts "there was an error with your input"
end

x > 5 ? unlock_door : display_error
```

Object Oriented Programming (OOP)

Creating a Class

```
class Robot
…
end
```

Creating Instances

```
class Robot
…
end

roomba = Robot.new
```

Setting Initial State

```
class Robot

  def initialize(name)
    @name = name
  end

end

roomba = Robot.new('roomba')
```

Instance Variables

```
class Robot

  def initialize(name)
    @name = name
  end

end

roomba = Robot.new('roomba')
```

Class Variables

```ruby
class Robot

  @@three_laws = ["don't harm humans",
  "obey orders", "protect yourself"]

  def initialize(name)
    @name = name
  end

  def recite_laws
    puts @@three_laws
  end
end
```

Accessors

```
class Robot
  def initialize(name, speed)
    @name = name
    @speed = speed
  end
end

r = Robot.new('roomba', 10)
puts r.name     # => error
puts r.speed    # => error

# we can't access name or speed  of
the robot from the instance
```

attr_reader

```
class Robot

  attr_reader :name, :speed

  def initialize(name, speed)
    @name = name
    @speed = speed
  end

end

r = Robot.new('roomba', 10)
puts r.name     # => 'roomba'
puts r.speed    # => 10
```

attr_writer

```ruby
class Robot
  attr_reader :name, :speed
  attr_writer :name
  def initialize(name, speed)
    @name = name
    @speed = speed
  end
end

r = Robot.new('roomba', 10)
r.name = 'roomba II'   # => 'roomba II'
```

attr_accessor

```ruby
class Robot

  attr_accessor :name, :speed

  def initialize(name, speed)
    @name = name
    @speed = speed
  end

end

r = Robot.new('roomba', 10)
# we can read and write to speed and name
```

Public Methods

```
class Robot

  def initialize(name, speed)
    @name = name
    @speed = speed
  end

  def move(direction)
   # move the robot
  end

end
```

Private Methods

```
class Robot

  …

  private
  def self_destruct
   # 3… 2… 1…
  end

end
```

Redefining Methods

```
class Robot

  def initialize(name, speed)
    @name = name
    @speed = speed
  end

  def move(direction)
   # move the robot
  end

end
```

Inheritance

```
class Robot

  attr_accessor :name, :speed

  def initialize(name, speed)
    @name = name
    @speed = speed
  end

  def move(direction)
   # move the robot
  end

end
```

```
class Roomba < Robot

  def start_vacuum()
   # start the vacuum
  end

end
```

Tripmeter