Ruby master - Feature #17380
Useful `include/prepend` in `refine`
12/09/2020 04:31 AM - marcandre (Marc-Andre Lafortune)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>matz (Yukihiro Matsumoto)</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
</tbody>
</table>

**Description**
Currently, prepend/include within a refine block leads to a method not being to see itself, or others defined in the same module:

```ruby
module Code
  def recurse(value = nil)
    return value if value
    recurse(42) # => NoMethodError!!!
  end
end

module Extension
  refine Object do
    include Code
  end
end

using Extension
:x.recurse(:y) # => :y (ok)
:x.recurse       # => NoMethodError, was hoping for 42
```

I find this unintuitive and not useful.

The conclusion of the current situation from [shugo (Shugo Maeda)](https://bugs.ruby-lang.org/users/shugo) and others is "I don't recommend module inclusion to define refined methods".

Could we change this situation so it can be recommended to use it?

What I believe would be more useful and is what I expected was that include/prepend within a Module would bring in the current methods in the Module, with the current refinements activated.

One use-case in particular is to publish libraries where one can give the option to the user to either:

- call using GreatExtension in each and every file that need it
- or MyClass.prepend GreatExtension once.

While [Jeremy Evans found a way to do it](https://bugs.ruby-lang.org/issues/17374#note-8), it remains challenging and unnatural.

**Related issues:**
Related to Ruby master - Bug #17374: Refined methods aren't visible from a refinement's module added

**History**

### #1 - 12/09/2020 06:40 PM - Eregon (Benoit Daloze)
- Related to Bug #17374: Refined methods aren't visible from a refinement's module added

### #2 - 12/09/2020 06:43 PM - Eregon (Benoit Daloze)
Maybe we should allow include RefinedImplementation from [https://bugs.ruby-lang.org/issues/17374#note-8](https://bugs.ruby-lang.org/issues/1734#note-8)?
Copying methods manually seems to have a very similar effect, but it would be more convenient.

### #3 - 12/10/2020 01:45 PM - Dan0042 (Daniel DeLorme)
It would be nice if prepend/include worked within a refine block, but if they don't then at least it should raise an error. In that respect I disagree with closing #17374; even if the result is "expected", the fact that including a module is effectively a no-op should be considered a bug. Silently failing to have an effect is not so good.
including a module is effectively a no-op

It isn't a no-op, as it does bring each method in the refinement, but those methods "live" outside of said refinement. See my example above where
:x.recurse(y) #=> y (ok).

Yes, I understand that, but even if the methods are technically in the refinement, if they are unreachable then effectively it's the same as a no-op. Although as the example shows it's more like a "half-op"; the methods are reachable from the outside but not the inside.