Ruby - Bug #20302

Multiple refinements cannot be applied to the same module

02/26/2024 10:10 AM - mame (Yusuke Endoh)

Status:	Closed		
Priority:	Normal		
Assignee:	shugo (Shugo Maeda)		
Target version:			
ruby -v:	ruby 3.4.0dev (2024-02-20T07:17:20Z master a605234bfa) [x86_64-linux]	**Backport:**	3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN

Description

If we are using multiple refinements for the Kernel module, it seems that only the last refinement is enabled.
In the following example, we can call f2 of M2, which is enabled later. but cannot call f1 of M1.

```
module M1
  refine(Kernel) do
    def f1 = :f1
  end
end

module M2
  refine(Kernel) do
    def f2 = :f2
  end
end

class Foo
  using M1
  using M2

  def test
    p f2 #=> :f2

    p f1 # expected => :f1
         # actual => undefined local variable or method 'f1' for an instance of Foo
  end
end

Foo.new.test
```

Note that if I change refine(Kernel) to refine(Object), it works as expected.
So I think there is a problem only when the refinement target is a module.

Associated revisions

Revision d50f9ca2dd416d92152cd958b5f39496088481b0 - 02/27/2024 05:51 AM - shugo (Shugo Maeda)

[Bug #20302] Multiple refinements cannot be applied to the same module

In the following code, the iclass tree of refinements in cref should be <iclass of Kernel@M2> -> <iclass of Kernel@M1> -> Kernel.

However, the iclass tree was broken because of code for included modules of refinements in rb_using_refinement().
Refinement#include is now removed, so this commit removes such unnecessary code.

```
module M1
  refine(Kernel) do
    def f1 = :f1
  end
end

module M2
  refine(Kernel) do
    def f2 = :f2
  end
end
```

```
class Foo
  using M1
  using M2

  def test
    p f2 #=> :f2

    p f1 # expected => :f1
         # actual => undefined local variable or method 'f1' for an instance of Foo
  end
end

Foo.new.test
```

Revision d50f9ca2dd416d92152cd958b5f39496088481b0 - 02/27/2024 05:51 AM - shugo (Shugo Maeda)

[Bug #20302] Multiple refinements cannot be applied to the same module

In the following code, the iclass tree of refinements in cref should be <iclass of Kernel@M2> -> <iclass of Kernel@M1> -> Kernel.

However, the iclass tree was broken because of code for included modules of refinements in rb_using_refinement().
Refinement#include is now removed, so this commit removes such unnecessary code.

```
module M1
  refine(Kernel) do
    def f1 = :f1
  end
end

module M2
  refine(Kernel) do
    def f2 = :f2
  end
end

class Foo
  using M1
  using M2

  def test
    p f2 #=> :f2

    p f1 # expected => :f1
         # actual => undefined local variable or method 'f1' for an instance of Foo
  end
end

Foo.new.test
```

Revision d50f9ca2 - 02/27/2024 05:51 AM - shugo (Shugo Maeda)

[Bug #20302] Multiple refinements cannot be applied to the same module

In the following code, the iclass tree of refinements in cref should be <iclass of Kernel@M2> -> <iclass of Kernel@M1> -> Kernel.

However, the iclass tree was broken because of code for included modules of refinements in rb_using_refinement().
Refinement#include is now removed, so this commit removes such unnecessary code.

```
module M1
  refine(Kernel) do
    def f1 = :f1
  end
end

module M2
  refine(Kernel) do
    def f2 = :f2
  end
end

class Foo
  using M1
  using M2
```

```
  def test
    p f2 #=> :f2

    p f1 # expected => :f1
         # actual => undefined local variable or method 'f1' for an instance of Foo
  end
end

Foo.new.test
```

History

#1 - 02/27/2024 05:51 AM - shugo (Shugo Maeda)

- Status changed from Open to Closed

Applied in changeset git|d50f9ca2dd416d92152cd958b5f39496088481b0.

[Bug #20302] Multiple refinements cannot be applied to the same module

In the following code, the iclass tree of refinements in cref should be <iclass of Kernel@M2> -> <iclass of Kernel@M1> -> Kernel.

However, the iclass tree was broken because of code for included modules of refinements in rb_using_refinement().
Refinement#include is now removed, so this commit removes such unnecessary code.

```
module M1
  refine(Kernel) do
    def f1 = :f1
  end
end

module M2
  refine(Kernel) do
    def f2 = :f2
  end
end

class Foo
  using M1
  using M2

  def test
    p f2 #=> :f2

    p f1 # expected => :f1
         # actual => undefined local variable or method 'f1' for an instance of Foo
  end
end

Foo.new.test
```