Add method marshalable?

Some objects can not be marshaled. Maybe there should be a method to tell it.

```ruby
hash = Hash.new { |h,k| k * 2 }
```

this hash can't be marshaled because it has a default proc. If existing such method:

```ruby
Marshal.marshalable?(hash) #=> method "Marshal.marshalable?"
hash.marshalable? #=> method "Kernel#marshalable?"
```

If you think the method name hard to spell, maybe get a synonym "dumpable?"

What would it do in this case?

```ruby
hash = Hash.new { |h,k| k * 2 }
[hash].marshalable?
```

So the method needs to recurse arbitrarily deep. So #marshalable will need to know just as much as Marshal.dump itself does about the object references contained in an object. Abstracting this out of Marshal.dump will be complex and will depend on external libraries and gems, which do custom dumping.

Why not just marshal and rescue?

Or:

```ruby
hash = Hash.new { |h, k| h[k] = k * 2 }
class << hash
def marshal_dump
  Hash[*self.to_a]
```
hash.marshatable? # => true

#3 - 11/09/2012 06:04 PM - mame (Yusuke Endoh)
- Target version set to 2.6

#4 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)
- Target version deleted (2.6)

#5 - 10/05/2019 11:38 PM - mame (Yusuke Endoh)
- Has duplicate Feature #16240: Are all objects and methods marshalable added