Class method and alternative constructor in classic Perl OOP

What if e would like to provide a user with an alternative way to instantiate an object? For example we would like to accept strings like '2013-11-23'?

We could somehow change the new method to notice when it only receives a single string and handle that case separately, but moving this into a separate method provides an opportunity to write cleaner code.

This is all the code we have to add to our Date.pm file from the example with stringification.

```
1.  # YYYY-MM-DD
2.  sub new_from_string {
3.      my ($class, $date_string) = @_;
4.
5.      my ($year, $month, $day) = split /-/, $date_string;
6.      return $class->new(year => $year, month => $month, day =>
    $day);
7.  }
```

Then we can write the following code:

```
1.  use strict;
2.  use warnings;
3.  use 5.010;
4.
5.  use FindBin;
6.  use File::Spec;
7.  use lib File::Spec->catdir($FindBin::Bin, '..', 'lib');
8.  use My::Date;
9.
10. my $d1 = My::Date->new(year => 2013, month => 1, day => 27);
11. say $d1;
12.
13. my $d2 = My::Date->new_from_string('2013-1-28');
14. say $d2;
```

and we get the output:

```
Date(2013, 1, 27)
```

```
Date(2013, 1, 28)
```

Class method

The new_from_string is called a class-method as it is called on the class (My::Date) instead of on a specific instance (e.g. $d). Compare My::Date->new_from_string with My::Date->new and with $d->year. For the first two, the value on the left side of the arrow is the name of the class, for the third example, the value on the left hand side of the arrow is the instance object.

When perl runs a method with this arrow notation, it always passes the value from the left hand side of the string as the first parameter. In the case of the above use of new and new_from_string this will be the name of the class: My::Date

The actual method does not do much. It just splits the string it received and the calls the real constructor with the values in the format it expects them.

Avoiding calling the class method as object method

There is nothing in the above code that would stop us from calling the class-method on an instance object like this:

```
1.  my $d3 = $d2->new_from_string('2001-2-3');
2.  say $d3;
3.  say $d3->year;
```

It will even work. Sort of. Incorrectly.

The output of the above code (if pasted at the end of the script earlier this article looks like this:

```
Date(2013, 1, 27)

Date(2013, 1, 28)

Date(2013, 1, 28)=HASH(0x7fba51827d58)

Can't locate object method "year" via package "Date(2013, 1, 28)" at
bin/date.pl line 19
```

The new_from_string returned a strange something Date(2013, 1, 28)=HASH(0x7fba51827d58), but calling year really threw an exception.

Part of this strangeness comes from the fact that we have overloaded the stringification. If comment out the 2 lines starting with use overload and run the script again we get a different output:

```
My::Date=HASH(0x7fb92904f300)

My::Date=HASH(0x7fb928827c38)

Attempt to bless into a reference at ../lib/My/Date.pm line 7.
```

The first two lines were the regular string representations of two My::Date object. The third line is the exception we got when we tried to call bless passing a reference instead of a string (the name of the class).

In general we'd better avoid this situation. We can change the new_from_string function to make sure the variable $class is a string and not a blessed object.

```
1.  use Carp qw(croak);
2.  use Scalar::Util qw(blessed);
3.
4.  # YYYY-MM-DD
5.  sub new_from_string {
6.      my ($class, $date_string) = @_;
7.
8.      croak("new_from_string must be called on a class-name")
9.          if blessed $class;
10.
11.     my ($year, $month, $day) = split /-/, $date_string;
12.     return $class->new(year => $year, month => $month, day =>
    $day);
13. }
```

```
Date(2013, 1, 27)

Date(2013, 1, 28)

new_from_string must be called on a class-name at bin/date.pl line 17.
```

It still throws an exception, but this time we decide on the text, and if we used croak instead of die the user would even see the line where she called the new_from_string method and not where we detected the problem inside our class.