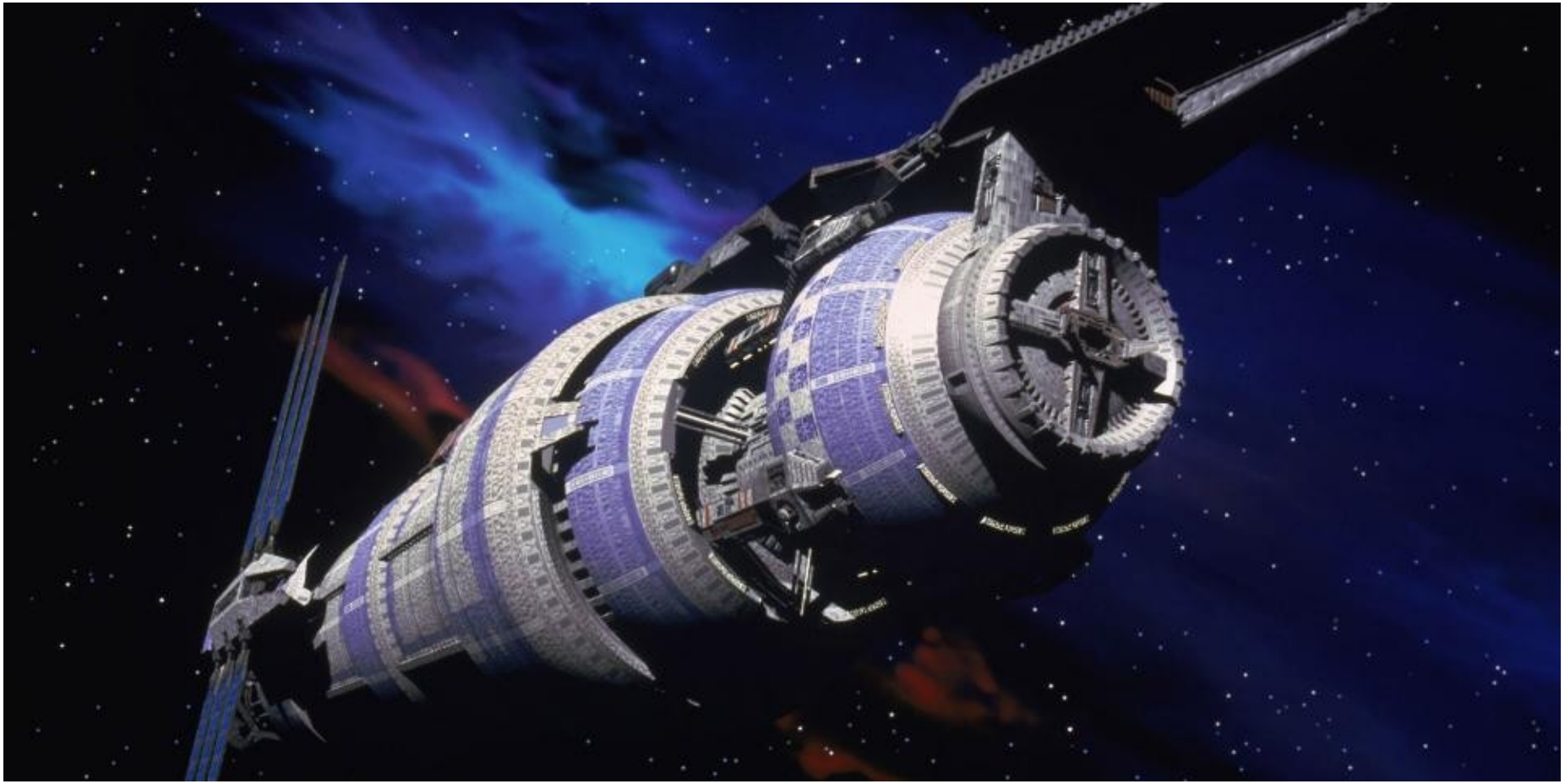
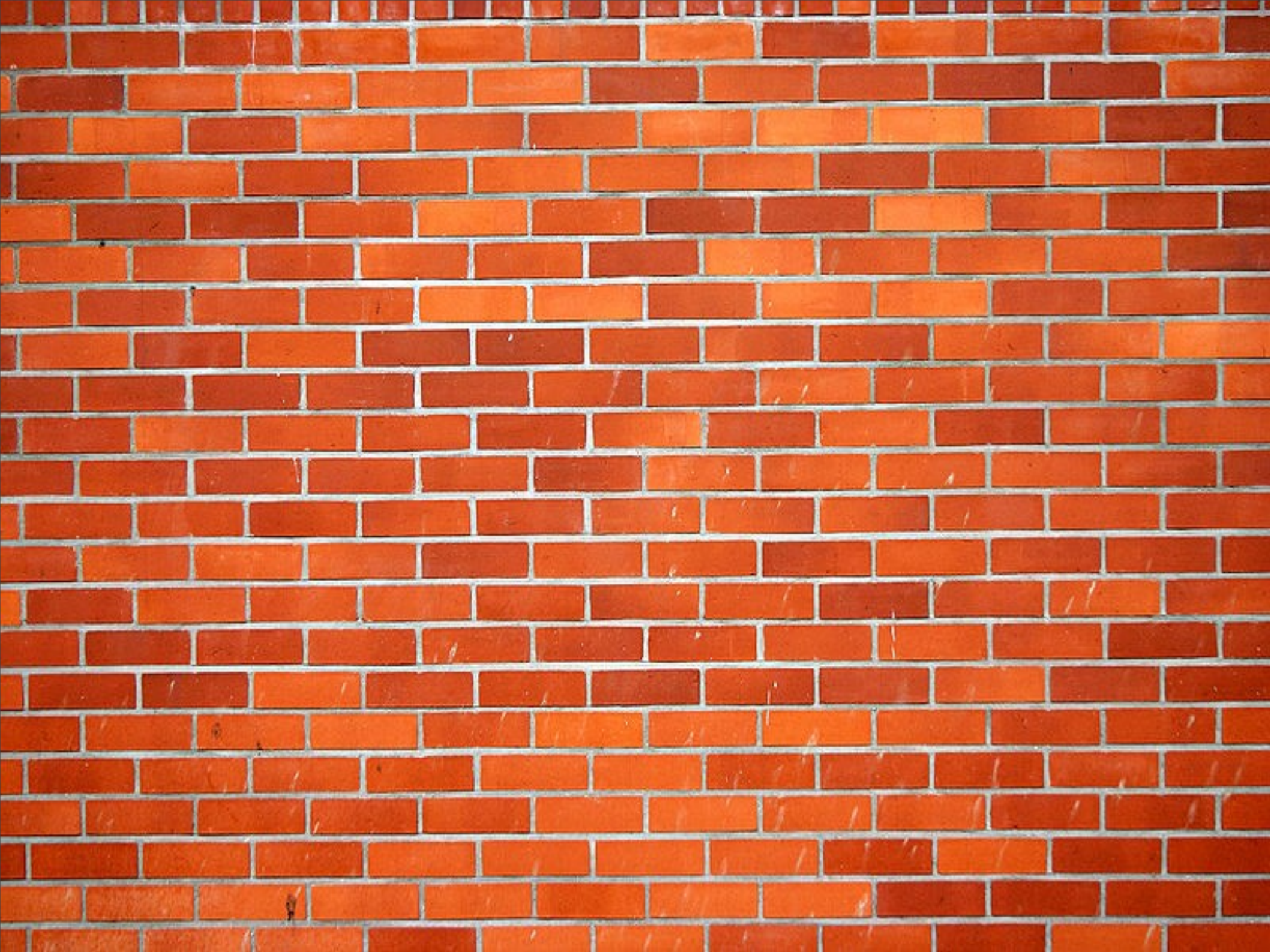

PROGRAMMING REPUBLIC OF
Perl
2013
www.perl.org
TUIT





Building Bridges

```
use NativeCall;
```

```
use NativeCall;

sub some_function(int)
    is native('libsomething')
    { * }

some_function(42);
```

libperl.so

Inline::Perl5

catalyst

Web Framework

```
#!/usr/bin/env perl

BEGIN {
    $ENV{CATALYST_SCRIPT_GEN} = 40;
}

use Catalyst::ScriptRunner;
Catalyst::ScriptRunner->run(
  'CiderWebmail',
  'Server'
);

1;
```

```
#!/usr/bin/env perl6

%*ENV<CATALYST_SCRIPT_GEN> = 40;

use Inline::Perl5;

my $p5 = Inline::Perl5.new;

$p5.run(q:heredoc/PERL5/);
    use lib qw(lib);
    use Catalyst::ScriptRunner;
    Catalyst::ScriptRunner->run(
        'CiderWebmail',
        'Server'
    );
PERL5
```

```
use Inline::Perl5;

%*ENV<CATALYST_SCRIPT_GEN> = 40;

my $p5 = Inline::Perl5.new;

$p5.use('lib', 'lib');
$p5.use('Catalyst::ScriptRunner');
$p5.invoke(
    'Catalyst::ScriptRunner',
    'run',
    'CiderWebmail',
    'Server'
);
```

```
use Inline::Perl5;
my $p5 = Inline::Perl5.new;

$p5.use('Petal');
my $template = $p5.invoke(
    'Petal',
    'new',
    'bars.xhtml'
);
say $template.process({
    city => 'Linz',
    bars => <Chelsea Walker Bugs>,
});
```

```
use Digest::MD5:from<Perl5>;

say md5('foo');
```

```
use XML::XPath:from<Perl5>;

my $xp = XML::XPath.new(
    'xml-xpath.xml'
);
my $nodeset = $xp.find('//baz/@qux');

say $nodeset.get_node(1).getData;
```


ROBOT

Bot::BasicBot example

```
use Bot::BasicBot:from<Perl5>;

class P6Bot is Bot::BasicBot {
   ...
}
```

Bot::BasicBot example

```
method said(%statement) {
    self.reply(
        $%statement,
        "Hullo {%statement<who>}!"
    ) if %statement<body> eq 'Hi bot!';

    self.shutdown('leaving...')
        if %statement<body> eq 'bot quit';
}
method can($method) {
    return 1;
}
```

Bot::BasicBot example

```
my $bot = P6Bot.new(
    server   => "irc.freenode.org",
    port     => "6667",
    channels => ["#perl6"],
    nick     => "p6basicbot",
    name     => "Yet Another Bot",
);

$bot.run;
```

HTML::Parser example

```
use HTML::Parser:from<Perl5>;

class PrettyPrinter is HTML::Parser {
    has $!level = 0;
    ...
    method indent() {
        return '    ' x $!level;
    }
}
```

HTML::Parser example

```
method start($tag, %attrs, @attrs, $full) {
    say self.indent
        ~ '<'
        ~ $tag
        ~ @attrs.map(
              { qq/$_="{%attrs{$_}}"/ }
          ).join
        ~ '>';
    $!level++;
}

method text($content, *@args) {
    say $content.indent(*).trim.indent($!level *
4);
}
```

HTML::Parser example

```
PrettyPrinter.new\
    .parse_file('html-parser.html');
```

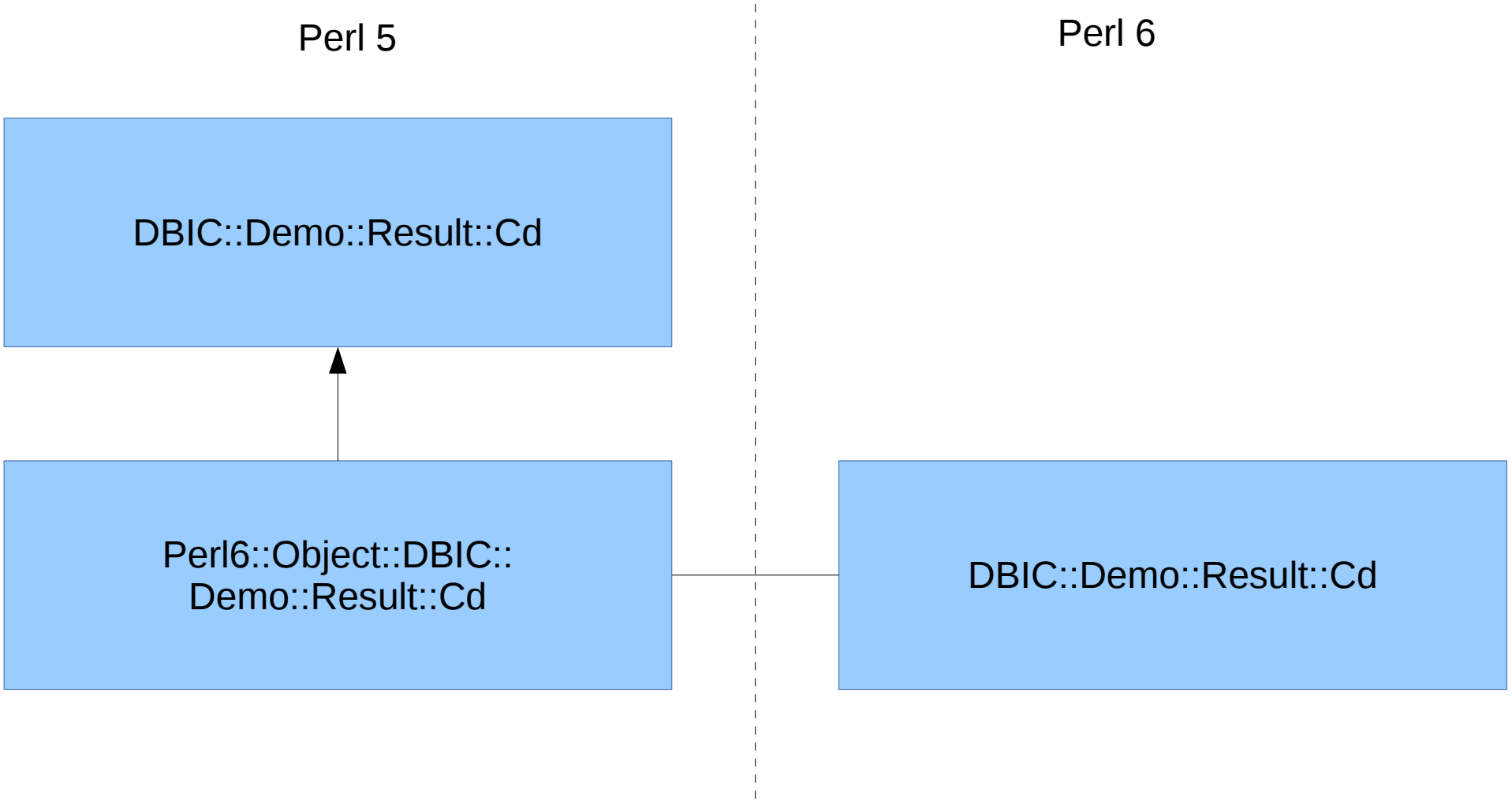
DBIx::Class

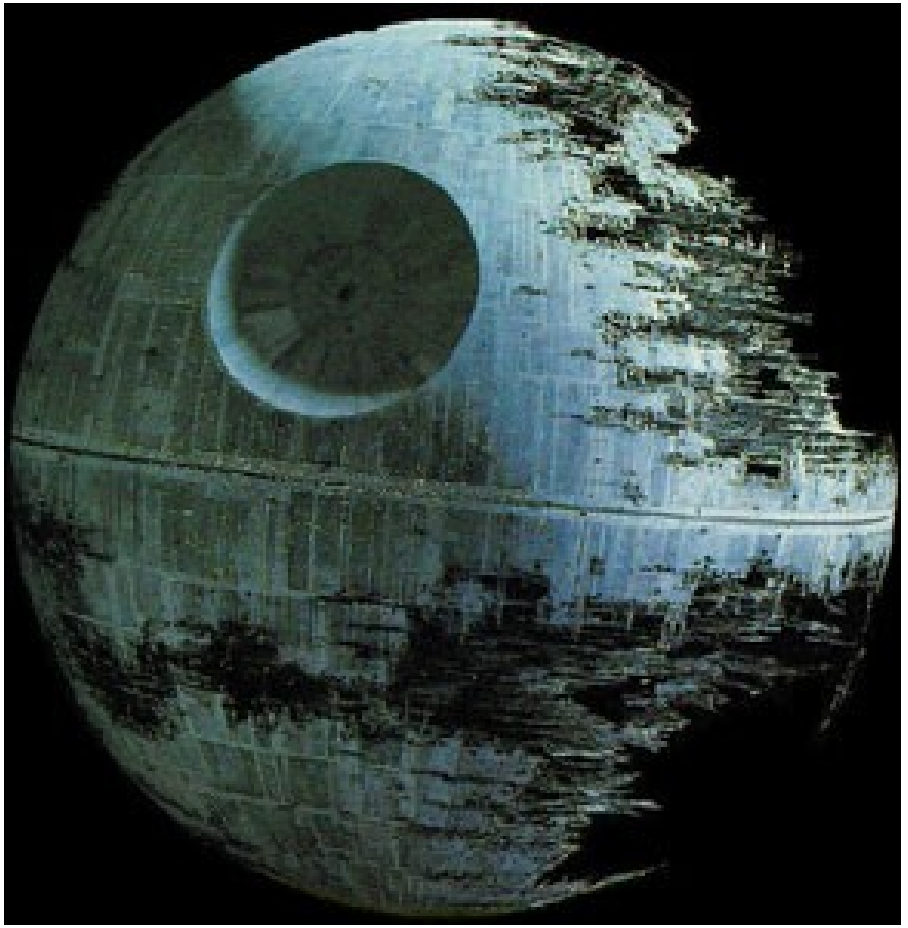
```
use DBIC::Demo:from<Perl5>;
my $schema = DBIC::Demo.connect('dbi:Pg:dbname=nine');

my $nin = $schema.resultset('Artist').create({
    name => 'Nine Inch Nails',
});
$nin.create_related('cds', {
    title => 'The Downward Spiral',
});

my $cd = $schema.resultset('Cd').search_rs.first;

say $cd.full_title;
```


CHALLENGE
ACCEPTED

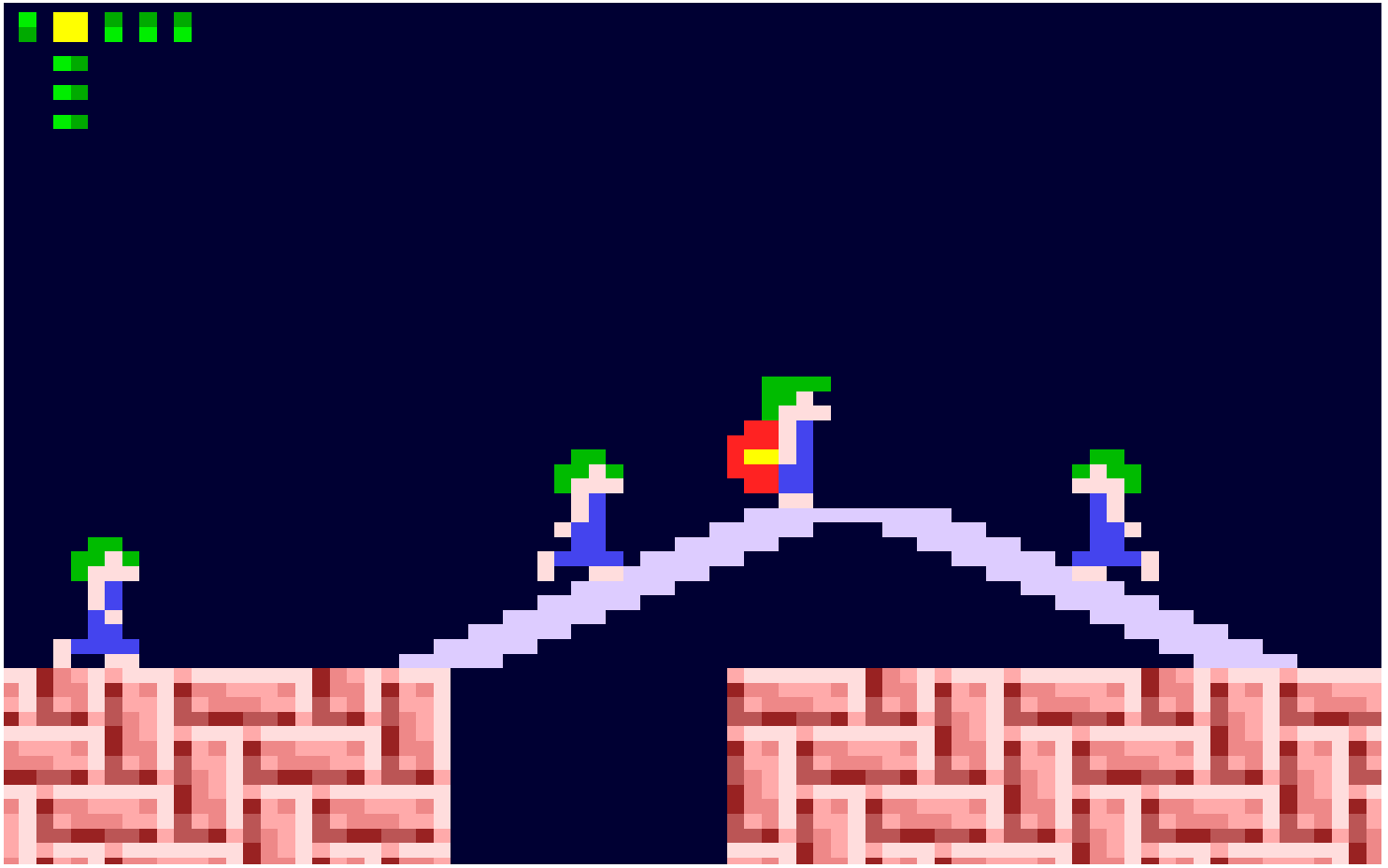
```
use utf8;
package DBIC::Demo::Result::Cd;
...
__PACKAGE__->belongs_to(...);

# Created by DBIx::Class::Schema::Loader v0.07043 @ 2015-09-01
13:18:52
# DO NOT MODIFY THIS OR ANYTHING ABOVE!
md5sum:7kPADFeDHuCnkZ2VDAhaZg

use v6::inline constructors =>
    [qw(new inflate_result)];

method full_title() {
   "{$.artist.name} - {$.title}"
}
```

v6::inline


Perl 5
Perl 6
DBIC::Demo::Result::Cd
Perl6::Object::DBIC::
Demo::Result::Cd
DBIC::Demo::Result::Cd



Inline::Perl6

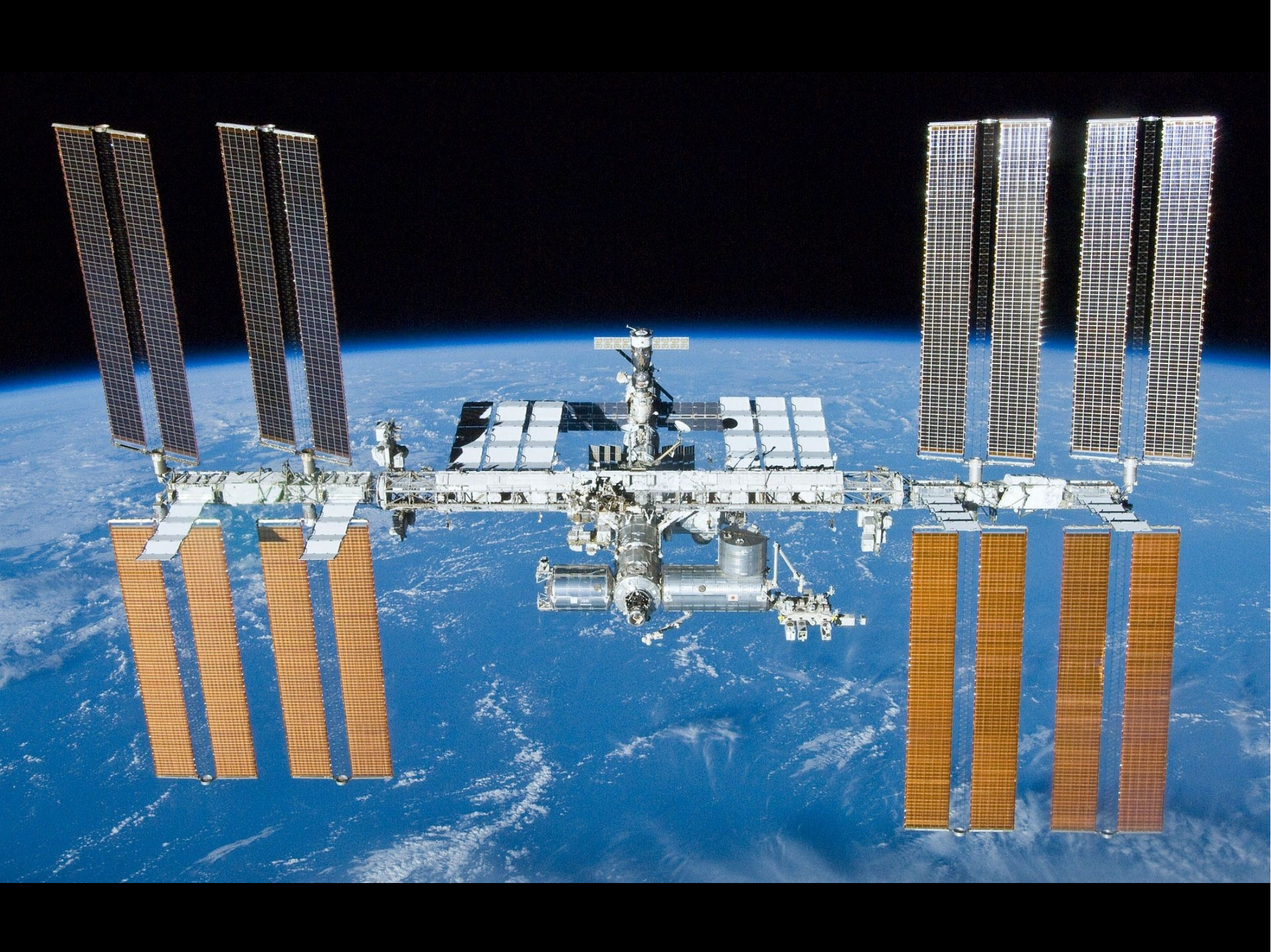


Inline::Perl6

```
use common::sense;
use Inline::Perl6;
BEGIN { Inline::Perl6::initialize };

v6::run('say <Hello from Perl6>');
v6::call('say', 'Hello from Perl6');

v6::run('
    class GLOBAL::MyStr {
        has $.value;
        method say() { $.value.say };
    }');
v6::invoke(
    'MyStr',
    'new',
    v6::named value => 'Hello from Perl6'
)->say;
```

Inline::Perl6

```
use common::sense;
use Inline::Perl6;
BEGIN { Inline::Perl6::initialize; }
use DBIC::Demo;

my $schema = DBIC::Demo->connect('dbi:Pg:dbname=nine');

my $nin = $schema->resultset('Artist')->create({
   name => 'Nine Inch Nails',
});

$nin->create_related('cds', {
   title => 'The Downward Spiral',
});

my $cd = $schema->resultset('Cd')->search_rs->first;

say $cd->full_title;
```



Thank You!

http://niner.name/talks/
http://github.com/niner/