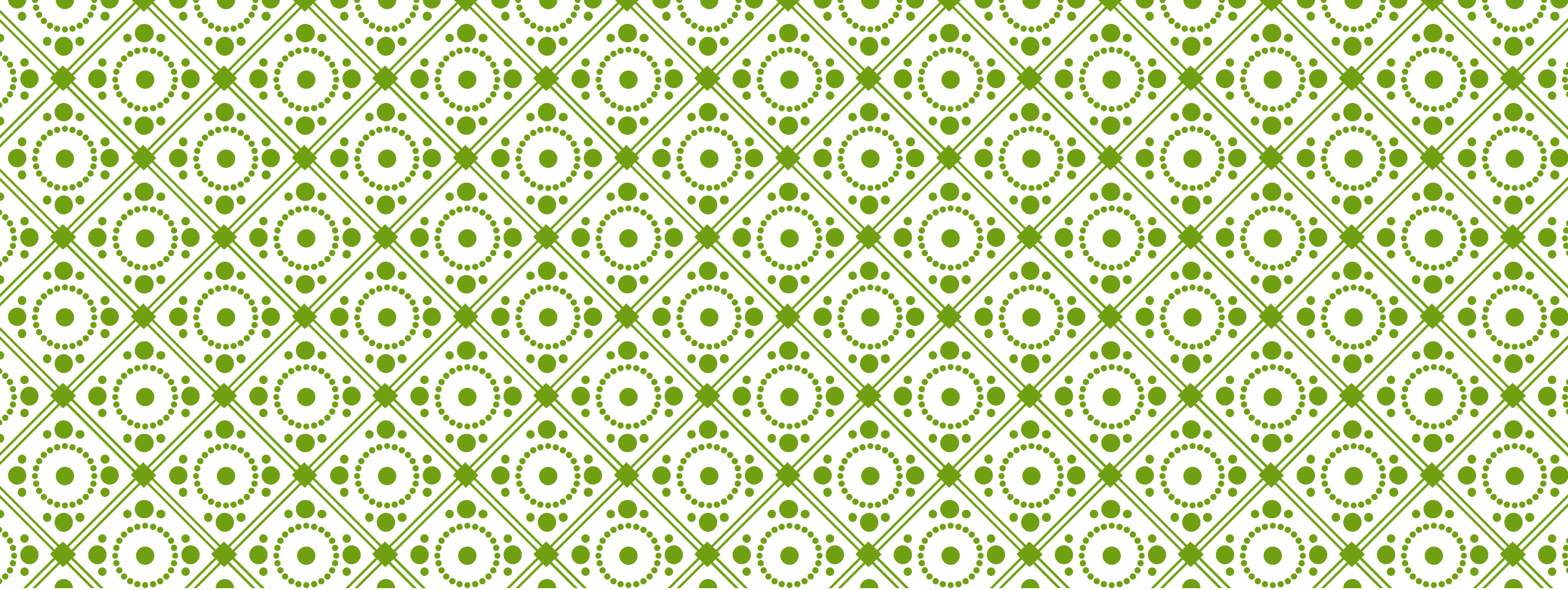


INTERFACES & ABSTRACT CLASSES

Andrew Sheehan

Boston University
Metropolitan College

WHAT IS AN INTERFACE

An Interface specifies methods that a class must implement

RULES ON ABSTRACT CLASSES

An abstract class cannot be created or instantiated

Just one (1) abstract method forces your class to become abstract

INTERFACES ARE PUBLIC

The methods and constant values must be declare public and have their body implemented in your class

ABSTRACT CLASSES
CAN USE NON-ABSTRACT FUNCTIONS

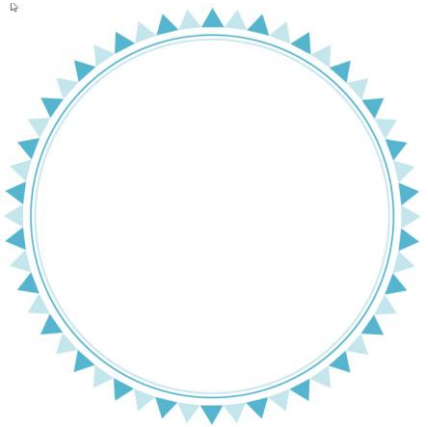
The methods and constant values must be declare public and have their body implemented in your class

ABSTRACT CLASSES CAN USE NON-ABSTRACT FUNCTIONS

```php
<?php
abstract class AbstractClass
{
    // Force Extending class to define this method
    abstract protected function getValue();
    abstract protected function prefixValue($prefix);

    // Common method
    public function printOut() {
        print $this->getValue() . "\n";
    }
}
?>
```

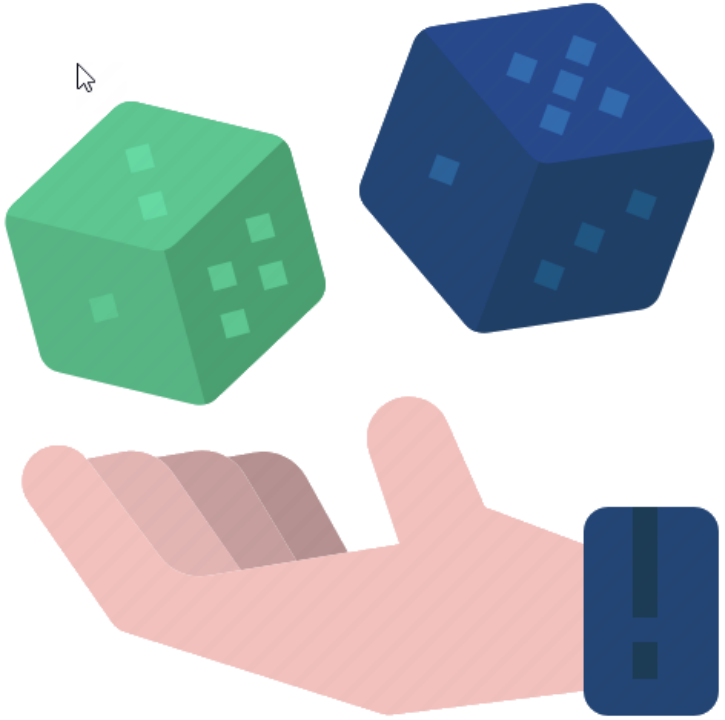
INTERFACE CHARACTERISTICS



- Methods with no implementations
- Methods will be public
- Classes can implement several interface, but only extend one (1) class.

INTERFACE
YES, CONSTRUCTOR

You can use a constructor in an interface in PHP – not typical..



EXAMPLE
INTERFACE

```php
<?php
    interface Command {
        public function execute();
    }
?>
```

EXTENDING INTERFACE

```php
<?php
    interface BatchCommand extends Command {
        public function  batch();
    }
?>
```

INTERFACE ADVANTAGES

- Allows unrelated classes to implement the same set of methods
- An interface can model multiple inheritances

Constants are Allowed.

INTERFACE CONSTANTS

```php
<?php
interface a
{
    const b = 'Interface constant';
}

// Prints: Interface constant
echo a::b;


// This will however not work because it's not allowed to
// override constants.
class b implements a
{
    const b = 'Class constant';
}
?>
```