https://youtu.be/ejJ8uoPtFoo

Operator Overloading

. In this exercise you'll see how to achieve polymorphism with **operator overloading**. You can choose any operator from the ASCII table and give it your own set of rules!

Operator overloading can be useful for many things. Consider the `+` operator. We can use it to add `int`s, `double`s, `float`s, or even `std::string`s.

In order to overload an operator, use the `operator` keyword in the function signature:

```
Complex operator+(const Complex& addend) {
  //...logic to add complex numbers
}
```

Imagine vector addition. You might want to perform vector addition on a pair of points to add their x and y components. The compiler won't recognize this type of operation on its own, because this data is user defined. However, you can overload the `+` operator so it performs the action that you want to implement.

Instructions

1. Define class `Point`.
2. Declare a prototype of overload method for `+` operator.
3. Confirm the tests pass.