

Mock Student Fee Provider

In this exercise, you will complete these tasks:



Write a collaborator interface.



A StudentRegistration has a IFeeProvider



Write tests using a mock object

Lab 4.2 - Overview

Write the collaborator interface,
with these methods:

```
void setFee( double fee);  
double feesDue();  
void payFee( double fee);
```

StudentRegistration
will use an IFeeProvider
to implement its student fee
calculation

- Use a mock object until we
get the "real"
implementation later

You need to "inject" the
collaborator

- Constructor(s) and/or setter
method



Lab 4.2 - Business Requirements

- A `StudentRegistration` sets an initial fee, pays the fee and gets the fees due

Now we'll implement the real business requirements, using our `IFeeProvider` collaborator

The `IFeeProvider` returns the fees due (via its `feesDue` method)

The `IFeeProvider` sets and collect payment (via the `setFee` and `payFee` methods, respectively)

Fees can set and paid at any amount

Paying the fees reduces the fees due

In your test class for StudentRegistration

Set up the mock object and put it in "playback" mode

- Set it up so that its methods can be called without any frequency or order constraints

Pass it into the business object (StudentRegistration)

- Which stores it in a field

What kind of mock should you use – regular, nice, or strict?

Lab 4.2 - Setup and Testing

Test StudentRegistration setFee,
payFee and feesDue functionality

- It's using the mock IFeeProvider collaborator now