

Mocks

1. Assuming InvoiceFilter is a business entity and IssuedInvoices is the DAO (data access object) connecting to a database, how would you go testing them (unit tests, mocks, integration tests, etc)?

```
1 public class InvoiceFilter {
2     private IssuedInvoices invoices;
3
4     public InvoiceFilter(IssuedInvoices invoices) {
5         this.invoices = invoices;
6     }
7     public List<Invoice> filter() {
8         return invoices.all().stream()
9             .filter(invoice -> invoice.getValue() < 100.0)
10            .collect(toList());
11    }
12 }
```

Integration tests for IssuedInvoices

Mock IssuedInvoices and unit test InvoiceFilter using the mock.

2. Consider the following class:

```
1 public class Geometry {
2     private Math math;
3
4     public Geometry(Math math) {
5         this.math = math;
6     }
7
8     public double distance(Point p1, Point p2) {
9         return math.sqrt(
10            math.pow(p2.getX() - p1.getX(), 2) + math.pow(p2.getY() - p1.getY(), 2));
11    }
12 }
```

Should you mock the Math class (see the following page)?

```

1 // Example test without mocking
2 @Test
3 public void testDistance1() {
4     var geometry = new Geometry(new Math());
5
6     Point p1 = new Point(0, 0);
7     Point p2 = new Point(3, 4);
8
9     assertEquals(5, geometry.distance(p1, p2));
10 }
11 // Example test with mocking
12 @Test
13 public void testDistance2() {
14     var mathMock = Mockito.mock(Math.class);
15     when(mathMock.pow(3)).thenReturn(109);
16     when(mathMock.pow(4)).thenReturn(116);
17     when(mathMock.sqrt(225)).thenReturn(5);
18
19     var geometry = new Geometry(mathMock);
20     Point p1 = new Point(0, 0);
21     Point p2 = new Point(3, 4);
22
23     assertEquals(5, geometry.distance(p1, p2));
24
25     verify(mathMock).pow(3);
26     verify(mathMock).pow(4);
27 }

```

Probably not a good idea to mock it because if the Math class is changed in the future in a way that is not backwards compatible then the test double is useless and needs to be updated.

3. Come up with as many advantages and disadvantages of mocking (stubs and spies including) as you can.

Answers may vary

Advantages:

- reduced dependency and increased control during testing
- separation of focus during testing

Disadvantages:

- may make the tests less realistic
- mocks may need to be maintained as the code they mock or is dependent on changes