

Port me if you can

Using ports & adapters


CONFIDENTIAL
CODE

Problem

```
class Mock {
public:
    Mock() {}
    Mock(int x) {}
    Mock(int x, int y) {}
    Mock(int x, int y, int z) {}
    Mock(int x, int y, int z, int w) {}
    Mock(int x, int y, int z, int w, int v) {}
    Mock(int x, int y, int z, int w, int v, int u) {}
    Mock(int x, int y, int z, int w, int v, int u, int t) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f, int e) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f, int e, int d) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f, int e, int d, int c) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f, int e, int d, int c, int b) {}
    Mock(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f, int e, int d, int c, int b, int a) {}
};

int main() {
    Mock m;
    Mock m1(1);
    Mock m2(1, 2);
    Mock m3(1, 2, 3);
    Mock m4(1, 2, 3, 4);
    Mock m5(1, 2, 3, 4, 5);
    Mock m6(1, 2, 3, 4, 5, 6);
    Mock m7(1, 2, 3, 4, 5, 6, 7);
    Mock m8(1, 2, 3, 4, 5, 6, 7, 8);
    Mock m9(1, 2, 3, 4, 5, 6, 7, 8, 9);
    Mock m10(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
    Mock m11(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11);
    Mock m12(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12);
    Mock m13(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13);
    Mock m14(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14);
    Mock m15(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15);
    Mock m16(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16);
    Mock m17(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17);
    Mock m18(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18);
    Mock m19(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19);
    Mock m20(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20);
}
```

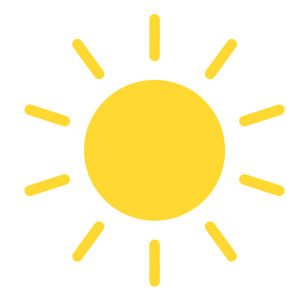



mock hell

Problem



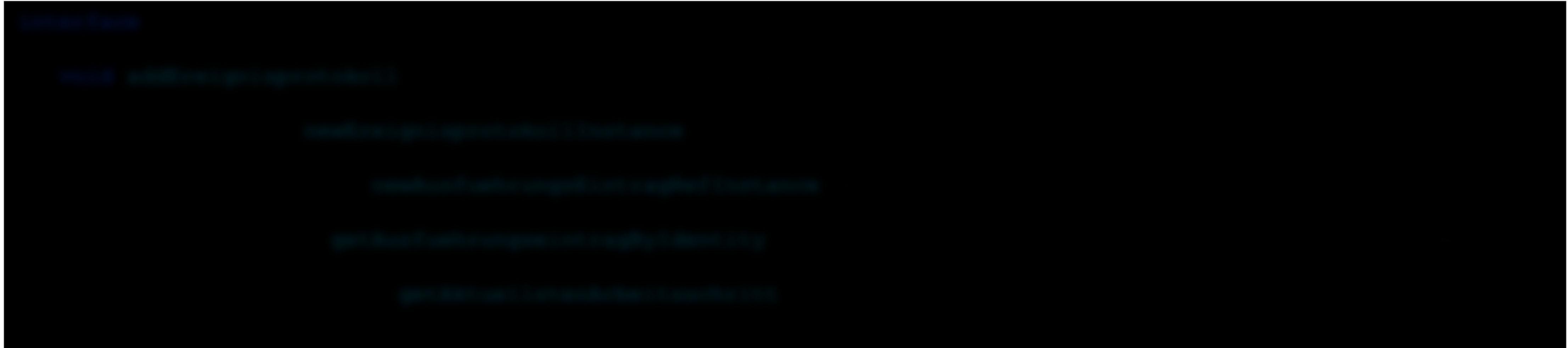
datapviderContext is our class (but in a different modul)
datapviderContext provides many functions
has many nested objects
I need just few operations, but on different nested objects
I have to know many internals on how datapviderContext is used
how can I reduce/simplify the mocking stuff?



treat datapviderContext as external and use ports/adapters to decouple

Solution

New interface (port) that defines only the operations I need



The implementation (adapter) just takes the original DataproviderContext as a constructor parameter and delegates all methods to it

The EreignisprotokollService gets the new Dataprovider interface injected (instead the old DataproviderContext)

```
dataprovider = mock(Dataprovider.class);  
ereignisprotokollService = new EreignisprotokollServiceNewImpl(dataprovider);
```

Solution

```
class
private
private
private

#initializeEach
void initializeEach
  @dataProvider = DataProvider.new
  @eventQueue = EventQueue.new
  @eventQueue.subscribe { |event| @dataProvider }
end

#Test
void test
  @dataProvider = DataProvider.new
  @eventQueue = EventQueue.new
  @eventQueue.subscribe { |event| @dataProvider }
end

private void initializeTest
  @dataProvider = DataProvider.new
  @eventQueue = EventQueue.new
  @eventQueue.subscribe { |event| @dataProvider }
end

private void initializeTest
  @dataProvider = DataProvider.new
  @eventQueue = EventQueue.new
  @eventQueue.subscribe { |event| @dataProvider }
end

private void initializeTest
  @dataProvider = DataProvider.new
  @eventQueue = EventQueue.new
  @eventQueue.subscribe { |event| @dataProvider }
end
```



mocking hell reduced to these lines



Conclusion

has reduced the complexity in the current implementation
testing was much easier
allowed me to define convenience methods to access legacy code

was he

Merçi for listening

and thank you for this great course

Contact:

Mail: juerg.weilenmann@css.ch

Twitter: -

Facebook: -

LinkedIn: -

Instagram: -

WhatsApp: -

Git: -

BeachBar: 18:00 - 20:00