

Example - Dice Cup

```
class DieCupTest {  
    DieCup testObj;  
  
    @Test  
    void initializeHasRightNumberOfDie() {  
        Die die1 = new Die(numSides: 6);  
        Die die2 = new Die(numSides: 12);  
        List<Die> diceToInject = new ArrayList<Die>();  
        diceToInject.add(die1);  
        diceToInject.add(die2);  
  
        testObj = new DieCup(diceToInject);  
  
        assertEquals(diceToInject.size(), testObj.dice.size());  
    }  
}
```

Initialization is easy enough...



```
public class DieCup {  
    List<Die> dice = new ArrayList<Die>();  
  
    public DieCup(List<Die> dice) {  
        this.dice = dice;  
    }  
}
```

Example - Dice Cup

```
public class FakeDie extends Die {  
    int weightedValue;  
  
    public FakeDie(int weightedValue, int numSides) {  
        super(numSides);  
        this.weightedValue = weightedValue;  
    }  
  
    @Override  
    public int getValue() {  
        return weightedValue;  
    }  
}
```

A now we have a deterministic unit test in isolation.

But...when we want to test calculating the total, we don't want to call the REAL die's roll method.

So...we use a FakeDie, via constructor injection.



```
@Test  
void rollSumsUpCorrectly() {  
    FakeDie fakeDie = new FakeDie(ARBITRARY_FAKE_ROLL, ARBITRARY_NUM_SIDES);  
    List<Die> diceToInject = new ArrayList<Die>();  
    diceToInject.add(fakeDie);  
    diceToInject.add(fakeDie);  
  
    testObj = new DieCup(diceToInject);  
    int value = testObj.roll();  
  
    assertEquals(ARBITRARY_FAKE_ROLL * 2, value);  
}
```