Verificação e Validação de Software I
Prof. Bernardo Copstein

Exercícios – Teste de classes com dependências

1) Mocks, stubs, e fakes. O que são e como eles diferem uns dos outros?

2) Como se pode melhorar a testabilidade da classe que segue? Demonstre o que deve ser alterado.

```
public class OrderDeliveryBatch {

  public void runBatch() {

    OrderDao dao = new OrderDao();
    DeliveryStartProcess delivery = new DeliveryStartProcess();

    List<Order> orders = dao.paidButNotDelivered();

    for (Order order : orders) {
      delivery.start(order);

      if (order.isInternational()) {
        order.setDeliveryDate("5 days from now");
      } else {
        order.setDeliveryDate("2 days from now");
      }
    }
  }
}

class OrderDao {
  // accesses a database
}

class DeliveryStartProcess {
  // communicates with a third-party webservice
}
```

3) A classe que segue deve calcular 15% de desconto no dia dos pais. Por que ela é difícil de testar? Como devemos modificá-la?

```
public class KingsDayDiscount {

  public double discount(double value) {
    Calendar today = Calendar.getInstance();
    boolean isKingsDay = today.get(MONTH) == Calendar.APRIL
       && today.get(DAY_OF_MONTH) == 27;
    return isKingsDay ? value * 0.15 : 0;
  }
}
```

4) A classe *InvoiceFilter* que segue é responsável por retornar as faturas de valor menor que 100. Ela usa a classe *IssuedInvoices* que é responsável pela comunicação com o banco de dados.

```
public class InvoiceFilter {
   private IssuedInvoices invoices;

   public InvoiceFilter(IssuedInvoices invoices) {
      this.invoices = invoices;
   }

   public List<Invoice> filter() {
      return invoices.all()
                        .stream()
                        .filter(invoice -> invoice.getValue() < 100.0)
                        .collect(toList());
    }
}
```

Qual das frases a seguir é falsa?

a) Testes de integração são a única forma de atingir 100% de cobertura de ramos.
b) A implementação usa injeção de dependências, o que habilita o uso de dublês.
c) É possível escrever testes unitários completamente independentes usando dublês.
d) A classe "IssuedInvoices" (da qual "InvoiceFilter" depende) deve ser testada no contexto de teste de integração.

5) Uma classe A depende de um método estático da classe B. Se queremos testar a classe A, qual das seguintes ações se aplica?
    a. Abordagem 1: criar um duble da classe B para o controlar o comportamento dos métodos da classe B.
    b. Abordagem 2: refatorar a classe A de maneira que o resultado do método da classe B seja recebido como parâmetro.

        i) Apenas a abordagem 1
        ii) Nenhuma das duas abordagens
        iii) Apenas a abordagem 2
        iv) Ambas

6) Considere as classes Calculadora, Somador e Subtrator fornecidas pelo professor:
    a. Refatore o código de maneira a atender os princípios de "design for testability" discutidos em aula.
    b. Crie um driver de teste unitário baseado em propriedades para as classes Somador e Subtrator
    c. Crie um driver de teste unitário para a classe Calculadora explorando Mocks.
    d. Crie um driver de teste de integração envolvendo todas as classes, agora sem usar os Mocks.

7) A classe "ServicoEstatisticas" calcula dois tipos de estatísticas sobre um conjunto de eventos: média, mediana e desvio padrão e maior ganho de performance. Para o cálculo da média, mediana e desvio padrão pode ser usadas abordagens diferentes. O cálculo do maior ganho de performance é fixo.

   Solicite ao professor o código destas classes e então faça o que se pede:

   a. Crie drivers de teste unitário para as classes *EstatisticaNomal* e *EstatisticaDesconsidera*. É necessário o uso de dubles neste caso? Que tipo de dubles?
   b. Crie um driver de teste unitário para a classe *ServicoEstatisticas*. É necessário algum tipo de duble neste caso? Que tipos de dubles?
   c. Crie um driver de teste para testar a integração da classe *EstatisticaNormal* com *ServicoEstatisticas*? É necessário algum tipo de duble neste caso? Que tipos de dubles?
   d. Crie um driver de teste para testar a integração da classe *EstatisticaDesconsidera* com *ServicoEstatisticas*? É necessário algum tipo de duble neste caso? Que tipos de dubles?

8) Análise a classe *CadastroProdutos* fornecida pelo professor. O que dificulta o teste desta classe? Altere a mesma de maneira que ela tenha uma interface mais adequada a testes. Crie um conjunto de casos de teste para a mesma e implemente o driver de teste.

9) Você está testando um sistema que dispara eventos avançados baseados em combinações complexas de condições boleanas externas relacionadas com o clima (temperatura externa, quantidade de chuva, vento e assim por diante). O sistema foi projetado de forma limpa e consiste em um conjunto de classes cooperativas, cada uma das quais tem uma responsabilidade única. Você usa teste baseado em particionamento para esta lógica e testa usando mocks. Qual das estratégias a seguir é válida?
   a. Usar mocks para substituir as condições observáveis externas:
   b. Criar diferentes objetos mock para cada variação que se deseja testar
   c. Usar mocks para controlar as condições externas e observar os eventos sendo disparados
   d. Usar mocks para controlar os eventos sendo disparados