Exame de Época Normal – 2017-07-04. Duração máxima: 105min.

Responda de forma objetiva às perguntas colocadas, em folha devidamente identificada.

Q1.

O "V-Model" e a metáfora da "pirâmide dos testes" (proposta por Mike Cohn) são duas formas de representar o processo de testes de software.

a) Apresente as principais ideias subjacentes a cada uma delas.
b) "A pirâmide dos testes é o V-Model invertido". Concorda com a afirmação? Justifique.

Q2.

As infraestruturas de partilha de código (SCM) baseadas em Git podem ser usadas segundo diferentes processos de trabalho (*workflow*), desde centralizados a totalmente distribuídos.

a) Qual o *workflow* que preconiza que cada programador tenha o seu próprio ramo de desenvolvimento (*branch*)? Justifique.
b) O Git suporta a revisão de código (*code review*) em equipa? Justifique, detalhando os fluxos associados a esta prática.

Q3.

Os testes que apresentam resultados intermitentes ("flacky tests") são um problema na infraestrutura de garantia de qualidade, porque podem levar toda a equipa a perder a confiança no sistema de Integração Contínua.

a) O que é o sistema de Integração Contínua e porque é importante que a equipa tenha confiança nele?
b) Que causas podem levar a que o sistema de integração contínua exiba "flacky tests"?
c) Como é que estratégias de mitigação de dependência podem contribuir para reduzir este problema? Justifique.

Q4.

"A utilização de uma ferramenta BDD, como o Cucumber, permite-nos exprimir os testes orientados à lógica do negócio, e não como programas orientados às tecnologias [de programação dos testes]" *In: fórum de discussão do "Agile Connect Aveiro"*.

a) Quem deve escrever os testes, numa abordagem BDD? Justifique.
b) O que é que o BDD acrescenta/muda em relação ao TDD, na inserção das práticas de qualidade no processo de engenharia de software?

Q5.

No contexto da *dashboard* de qualidade criada pela ferramenta Jenkins, explique a utilização dos conceitos:

a) "Build"
b) "Job"
c) "Test result trend"

Q6.

"Recomendação: fornecer contexto nas exceções. Cada exceção que é lançada deve fornecer contexto suficiente para determinar a origem e a localização de um erro. [...] O *stacktrace* não pode indicar a intenção da operação que falhou. Crie mensagens de erro informativas e passe-as junto com suas exceções." In: Martin, R. C. (2009). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.

a) O Java distingue entre exceções "checked" e "unchecked". Qual a diferença? Quais as que estão implicadas na recomendação citada?
b) Apresente práticas erradas ("anti-patterns") observadas no tratamento de erros com Exceções, em Java.

Q7.

O excerto de código junto apresenta um teste de integração para Java EE.

a) A anotação @RunWith [linha 31] passa o controlo do teste ao *framework* CukeSpace. Em que situações/para quê recomendaria a utilização do CukeSpace?

b) Há testes baseados no Arquillian que utilizam a anotação **@**RunAsClient. Seria adequado para este exemplo ou não? Justifique.

c) Qual a origem e funcionalidade da operação "assertThat" [linha 61]?

```
31  @RunWith(CukeSpace.class)
32  @Features({"src/test/resources/it/feature/date_conversion.feature"})
33  public class DateConversionFeatureIT {
34
35      @Deployment
36      public static JavaArchive createArchiveAndDeploy() {
37          return ShrinkWrap.create(JavaArchive.class)
38                  .addClasses(LocaleManager.class, TimeService.class)
39                  .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
40      }
41
42      @Inject
        TimeService timeService;
44
45      User user;
46      Date rawDate;
47
48      @Given("^a user named '(.+)'$")
49      public void create_user_with_name(final String name) throws Throwable {
50          user = new User(name);
51      }
52
53      @When("^this user enters the date '(.+)' into the time conversion service$")
54      public void user_enters_the_date(@Format("yyyy-MM-dd HH:mm:ss") final Date date)
55              throws Throwable {
56          rawDate = date;
57      }
58
59      @Then("^the service returns a conversion hint with the message '(.*)'$")
60      public void service_returns_a_converted_date(final String dateConverted) throws Throwable {
61          assertThat(timeService.getLocalizedTime(rawDate, user), equalTo(dateConverted));
62      }
63  }
```

Q8.

Considere o trecho a seguir fornecido como o conteúdo do ficheiro *search_book.feature*

A partir deste exemplo, e tendo presente as construções do *framework* de testes Cucumber-JVM, demonstre (em código) a implementação dos testes unitários associados.

Declare as assunções que entendam necessárias para ajudar a perceber a sua implementação.

```
Feature: Book search
  To allow a customer to find his favorite books quickly, the library must offer multiple ways to search for a book.

 Scenario: Search books by publication year
   Given a book with the title 'One good book', written by 'Anonymous', published in 14 March 2015
    And another book with the title 'Some other book', written by 'Tim Tomson', published in 23 August 2016
    And another book with the title 'How to cook a dino', written by 'Fred Flintstone', published in 01 January 2012
   When the customer searches for books published between 2015 and 2016
   Then 2 books should have been found
    And Book 1 should have the title 'Some other book'
    And Book 2 should have the title 'One good book'
```