P2 – CES-28

Nome: Hudson William Braga Vieira

As respostas estão destacadas de vermelho.

QUESTÃO 2 - TDD

Dada as questões abaixo, marque a opção correta:

a) Ha um conjunto de requisitos e casos de teste a serem implementados em um projeto TDD. Durante o ciclo do TDD, percebemos que ha um caso limite ou exceção simples de implementar, que não foi previsto, e é importante e necessário para que tudo funcione. O que fazemos? (0.5)

   I. Não podemos adicionar funcionalidades, mesmo que pequenas. Precisamos recombinar as funcionalidades e casos previstos com o chefe ou o resto do grupo.

   II. Adicionamos mais um caso de teste na lista "to do", e o TDD continua normalmente. Esse novo teste será implementado como qualquer outro durante a fase RED do TDD.

   III. Necessariamente precisamos incluir esse caso nos testes já existentes, através de asserts extra.

   IV. Implementamos esse novo caso na fase BLUE do TDD, ou seja, considerando-o como uma refatoração, pois não existe teste para ele.

b) O TDD indica que não misturemos as fases RED, GREEN, e BLUE, e que hajam varias iterações do mesmo ciclo. Um dos propósitos disso eh fazer com que a qualquer momento no ciclo TDD, apenas um (o novo teste) ou pelo menos, poucos testes estejam falhando ou com erro. Ou seja, fazer com que o código de produção permaneça relativamente próximo de "compilando, executando e 100% *green tests*", mesmo durante o desenvolvimento. A não ser em casos raros em que um bug ou refatoração muito fundamentais quebrem momentaneamente muitos testes, gostaríamos que isso fosse sempre verdade. Consideramos que isso ajuda a manter a qualidade do software no TDD. (0.5)

   I. V (X)

   II. F ( )

c) Antes de implementar um modulo de software em TDD, listamos uma lista de funcionalidades e/ou casos de teste. Começamos a implementação destas funcionalidades, e no meio da lista, percebemos que a implementação seria mais fácil e incremental se trocássemos a ordem das funcionalidades da lista. Mas o TDD não permite mudar a ordem preestabelecida a não ser que se renegociem as prioridades desses requisitos com o resto do grupo ou o chefe. (0.5)

   I. V ( )

II. **F (X)**

d) Assinale os itens que são verdadeiros: (0.5)

I. Testes funcionam como documentação, porque o teste é um exemplo de uso da classe.

II. O TDD exige que a única documentação seja na forma de testes, desde o começo até o produto final. Não vale nenhum documento escrito ou on-line, nem nenhuma ferramenta de organização.

III. **Um teste pode funcionar como documentação, mas para isso, precisa ser bem organizado. Separar testes correlatos em classes separadas, usar nomes descritivos para os métodos @Test, separar as fixtures em @Before ou @BeforeClass, etc, Essas práticas auxiliam os testes a serem legíveis e se comportarem como boas documentações.**

IV. Qualquer documentação pode incluir também exemplos de uso bem organizados. Uma vantagem de usar testes ao invés de documentação texto, é que se os testes são mantidos "green", se garante que os testes são atualizados quando o código muda, enquanto um exemplo em texto pode não funcionar mais.

V. O TDD exige que todos o comportamento da classe, aos mínimos detalhes, incluindo exatamente como se comportar em todos os casos limite e possíveis exceções, estejam bem especificados antes de se começar a programar, mesmo que estejam escritos no papel de forma mais informal, como lista de testes.

VI. Durante a fase de refatoração, é obrigatório focar apenas na nova funcionalidade implementada durante as ultimas fases RED-GREEN. Não podemos refatorar nada que não seja diretamente relacionado a nova funcionalidade.