1.1) Maria está trabalhando em um projeto de desenvolvimento de software e decide adotar a abordagem do Test Driven Development (TDD) para melhorar a qualidade do código e a colaboração entre os membros da equipe. Ela começa a implementar uma nova funcionalidade seguindo os princípios do TDD. Assinale a alternativa que apresenta corretamente o processo que Maria deve seguir

A. **Escrever o teste, escrever o código, executar o teste, refatorar e, por fim, integrar o código ao repositório principal. (X)**

B. Escrever o código, executar o teste, escrever o teste, refatorar e, por fim, integrar o código ao repositório principal.

C. Escrever o teste, executar o teste, escrever o código, executar o teste novamente, refatorar e, por fim, integrar o código ao repositório principal.

D. Escrever o código, escrever o teste, executar o teste, refatorar e, por fim, integrar o código ao repositório principal.

E. Escrever o teste, escrever o código, executar o teste, integrar o código ao repositório principal e, por fim, refatorar.

1.2) Justifique a resposta

O TDD é caracterizado por ciclos curtos de desenvolvimento, nos quais um teste é criado antes de implementar uma funcionalidade. Inicialmente, o teste recém-criado falha, pois a funcionalidade ainda não existe. Posteriormente, a funcionalidade é desenvolvida de forma a fazer o teste passar com sucesso. Isso garante um fluxo de desenvolvimento orientado pela criação de testes antes da implementação, como abordado na questão A.

2.1) Como você dividiria uma Epic em várias User Stories? Dê um exemplo.

- Defina objetivos gerais.
- Liste e priorize requisitos gerais.
- Identifique funcionalidades-chave independentes.
- Divida as funcionalidades com base em fluxos de trabalho ou jornadas do usuário.
- Estabeleça critérios de aceitação claros.
- Se necessário, divida em histórias de usuário e priorize.
- Acompanhe o progresso e ajuste conforme necessário.

Exemplo:
Epic: Desenvolvimento de um aplicativo de lista de tarefas.

1. Adicionar Tarefa: Os usuários podem inserir tarefas na lista.
2. Marcar Tarefa como Concluída: Os usuários podem sinalizar tarefas como concluídas.
3. Remover Tarefa: Os usuários podem excluir tarefas da lista.
4. Editar Tarefa: Os usuários podem modificar o nome de tarefas já existentes.
5. Priorizar Tarefas: Os usuários têm a capacidade de rearranjar a ordem das tarefas na lista.

2.2) Quais são os critérios de aceitação e por que são importantes em User Stories?

- Adição de Tarefa: Permite a inclusão de tarefas na lista, que devem ser visíveis sem estar marcadas como concluídas inicialmente.

- Marcação de Tarefa como Concluída: Fornece a opção de sinalizar tarefas como concluídas, identificando-as visualmente e tornando-as não editáveis.

- Remoção de Tarefa: Permite a exclusão de tarefas da lista sem afetar outras tarefas.

- Edição de Tarefa: Possibilita a modificação do nome de tarefas por meio de um clique duplo, com a capacidade de salvar as alterações.

- Priorização de Tarefas: Permite a reorganização de tarefas arrastando e soltando, mantendo a ordem desejada pelo usuário na lista.

3.1 O que é um teste unitário e qual é o seu objetivo?

Um teste unitário avalia a menor parte testável de um programa, que pode ser uma função em linguagens funcionais ou um método em linguagens orientadas a objetos. Seu propósito principal é confirmar que essa unidade de código opera corretamente, assegurando sua precisão e identificando erros precocemente, o que, por sua vez, contribui para melhorar a qualidade e facilidade de manutenção do software.

3.2) Explique o conceito de "mocking" em testes unitários.

Nos testes unitários, o "mocking" envolve criar objetos simulados, conhecidos como "mocks", que imitam o comportamento de partes reais do sistema. Isso é feito para isolar a unidade de código em teste.

4.1) O que é refatoração e como ela se encaixa no XP?

A refatoração é a ação de reorganizar o código-fonte de um software sem alterar sua funcionalidade externa. Seu objetivo é aprimorar a qualidade do código, tornando-o mais legível, mantível e eficiente, sem introduzir novos recursos ou corrigir erros. A refatoração é essencial no desenvolvimento de software, pois ajuda a evitar o acúmulo de problemas técnicos, mantendo o código limpo e robusto ao longo do tempo.

Dentro do contexto do Extreme Programming (XP), uma metodologia ágil de desenvolvimento de software, a refatoração desempenha um papel fundamental. Ela se alinha com os princípios do XP, como simplicidade, feedback rápido, melhoria contínua, redução da dívida técnica e colaboração, pois incentiva a equipe a trabalhar em conjunto para aprimorar o código.

4.2) Explique o conceito de "programação em pares" e seus benefício

A "programação em pares" envolve dois programadores trabalhando juntos em um computador. Isso melhora a qualidade do código, promove o aprendizado, a criatividade e a resolução eficaz de problemas, além de compartilhar tarefas e manter o foco. É uma prática que resulta em desenvolvimento de software mais eficiente e de maior qualidade.

4.3) Como o TDD (Test-Driven Development) é aplicado no XP?

No Extreme Programming (XP), o TDD (Desenvolvimento Orientado por Testes) é aplicado escrevendo testes antes de criar o código de implementação. Os testes são executados inicialmente e falham, pois a funcionalidade ainda não está implementada. A implementação mínima é feita para fazer os testes passarem, seguida pela refatoração. Esse processo é repetido para cada nova funcionalidade. O TDD no XP oferece benefícios como testes automatizados, documentação viva, detecção precoce de problemas, simplicidade, clareza e confiança nas mudanças de código.

5.1) Quais métricas você usaria para avaliar o sucesso de uma implementação de Scrum ou XP?

Para avaliar o sucesso de uma implementação de Scrum ou XP, eu usaria métricas como velocidade da equipe, cumprimento de compromissos, qualidade do software, satisfação do cliente, tempo de entrega, resultados de retrospectivas, nível de engajamento da equipe, redução de dívida técnica, ciclos de feedback e estabilidade do processo. Eu escolheria métricas alinhadas aos objetivos da implementação e do negócio, garantindo um equilíbrio entre produtividade e qualidade.

5.2) Explique como User Stories e Epics podem ser usadas para facilitar a comunicação entre desenvolvedores e stakeholders não técnicos

User Stories e Epics são ferramentas de comunicação que ajudam a facilitar o entendimento entre desenvolvedores e stakeholders não técnicos em projetos de software. User Stories se concentram em necessidades do usuário, têm linguagem simples e critérios de aceitação mensuráveis. Epics agrupam funcionalidades relacionadas, comunicam objetivos gerais e podem ser alinhadas com metas estratégicas. Ambos fornecem transparência, priorização e um quadro claro do progresso do projeto.