A picture containing icon

Description automatically generated

**Trabalho Pratico Engenharia de software**

Engenharia de software

Licenciatura em Tecnologias da informação

Pedro António Marçal Marcos Sampaio

Águeda | 17 de novembro de 2024

Resumo

Este projeto resume-se a um programa ao qual será passado como argumentos números e em certos casos específicos esses números serão substituídos por palavras. Os número e condições especificas para a substituição são as seguintes:

Caso o número fornecido seja divisível por 3, adicionar a palavra “Pim”, caso o número seja divisível por 5, adicionar a palavra “Pam” e finalmente, caso seja divisível por 7, será adicionada a palavra “Pum”. Adicionalmente caso o número contenha os dígitos 3,5,7, acrescentar “Pim”, “Pam”, “Pum”, respetivamente.

O desenvolvimento da solução para o problema anterior devera seguir uma metodologia de desenvolvimento seguida por testes, tanto testes unitários como refactoring.

O código deve ser simples e capar de ser entendido por qualquer programado, sem que seja necessário recorrer a comentários.

Índice Geral

[1. Introdução 1](#_Toc86923659)

[2. A ESTRUTURAR DE ACORDO COM AS ATIVIDADES DESENVOLVIDAS 2](#_Toc86923660)

[3. Apresentação e análise dos resultados 3](#_Toc86923661)

[4. Conclusões 4](#_Toc86923662)

1. Introdução

O desenvolvimento de software orientado por testes (Test-Driven Development - TDD) é uma metodologia que coloca os testes no centro do processo de construção do código, promovendo a criação de soluções mais robustas e alinhadas aos requisitos. Este relatório apresenta a aplicação desta abordagem no desenvolvimento de um programa em Java que, para um número dado, segue regras específicas para gerar uma representação textual.

O problema em questão exige que o programa substitua números por palavras com base na sua divisibilidade por 3, 5 e 7, e também pela presença dos dígitos 3, 5 e 7 no número. Para cada condição satisfeita, palavras como "Pim", "Pam" e "Pum" devem ser adicionadas à saída, ou, caso nenhuma regra seja aplicável, o próprio número deve ser devolvido.

Neste contexto, o objetivo foi implementar o método principal de forma simples e eficiente, assegurando que este cumpre os requisitos através da construção de uma bateria de testes unitários. Além disso, foi dada especial atenção à legibilidade do código e à sua capacidade de ser facilmente compreendido por outros programadores, como é promovido pelas metodologias ágeis.

O presente relatório documenta as estratégias de desenvolvimento e teste utilizadas, descreve os cenários abordados e justifica as decisões tomadas ao longo do processo.

1. Implementação

2.1- Primeiro esboço

Após receber o trabalho a primeira coisa que pensei em fazer, foi um esboço de como seria o projeto. Então rapidamente abri o Intelij e comecei a desenhar um esboço de como seria e como funcionaria o projeto, isto sem levar em conta os testes nem nada, apenas um simples e sem ser necessário nenhum comentário implementação em intelij.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 1- Primeira implementação básica

2.2. Implementação de testes

Comecei por criar um projeto em Maven, onde adicionei uma classe Main ao src de Java

E uma classe MainTest.

Para criarmos este programa seguindo um desenvolvimento de testes unitários, teremos de criar os testes primeiro e só depois de vermos que funciona, adicionar ao código.

2.2.1.-Teste a número sem divisibilidade ou que sejam apenas dígitos.

O primeiro teste testa números sem divisibilidade ou que sejam apenas dígitos e testei os numero dados pelo professor como exemplos:

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 2- Primeiros testes

Apos implementar o código para os primeiros testes, voltei a classe main, vazia até então e adicionei o código a ser testado.

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

Figura 3- Código inicial a ser testado

Todos os testes correram sem erro algum!

Uma imagem com texto, Software de multimédia, software, captura de ecrã

Descrição gerada automaticamente

Figura 4-Testes passados

2.2.2- Testes de divisibilidade

Agora criei testes para testar caso em que os números sejam divisíveis por 3, 5 e 7

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura 5-Testes de Divisibilidade por 3,5,7

Apos a criação dos testes irei novamente ao main e modifico o código de modo a poder testar!

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura 6-Código das divisões por 3,5,7

Após testar o programa, todos os testes, incluindo os anteriores aparentam estar corretos!

Uma imagem com texto, software, Software de multimédia, captura de ecrã

Descrição gerada automaticamente

Figura 7- Testes realizados com sucesso

2.2.3-Teste de Dígitos específicos

Estes testes verificam se os dígitos específicos 3, 5 e 7 estão presentes no número dado. Caso estejam, devem ser substituídos por “Pim”, “Pam” e “Pum” respetivamente.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 8-Testes para os dígitos 3,5,7

Apos criar os testes, adicionei o código á classe main.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura 9-Código quase finalizado

Apos adicionar este código ao main testei o programa e como veremos a seguir passou todos os testes até ao momento.

Uma imagem com texto, Tipo de letra, Software de multimédia, captura de ecrã

Descrição gerada automaticamente

2.2.4- Testes de combinações e casos extremos

Para finalizar os testes e para garantir uma melhor cobertura irei fazer testes a todos os exemplos dados na folha de exercício e a mais algumas combinações e casos que ache uteis.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura 10-Teste aos exemplos dados pt1

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 11-Teste aos exemplos dados pt2

Apos estes testes estarem escritos, mandei o código correr e como podemos ver todos correram corretamente.

Uma imagem com texto, software, Software de multimédia, Software gráfico

Descrição gerada automaticamente

Agora adicionarei testes para casos mais extremos.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 12-Exemplos de testes extremos

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 13-Programa passou a todos os testes

1. Apresentação e análise dos resultados

**Análise dos Resultados**

* **Comportamento Esperado:**  
  Os resultados obtidos estão em conformidade com os requisitos do problema. Em todos os casos, o programa retorna o resultado correto, fazendo o processamento das divisões e verificando as presenças dos dígitos pedidos.
* **Cobertura Abrangente:**  
  Os testes cobriram uma ampla variedade de cenários, incluindo divisões simples, presença dos dígitos 3, 5, 7, combinações de ambos e situações extremas como números múltiplos dígitos relevantes. Nenhuma inconsistência foi encontrada.
* **Qualidade do método:**  
  O método obterResultado mostrou-se robusto, com um desempenho correto em todas as condições testadas, confirmando a implementação lógica esperada.

Os testes demonstraram que o programa cumpre os requisitos especificados, garantindo precisão e consistência em todos os cenários testados. Através desta análise, podemos afirmar que o método está adequado e pronto para ser utilizado em produção.

1. Conclusões

Em conclusão, o desenvolvimento do programa dado no exercício demonstrou ser um bom exercício para praticar o método ágil de desenvolvimento orientado por testes. O método criado “obterResultado” foi projetado de maneira simples, eficiente e compreensível para todos os programadores sem ser necessário o uso de comentários. A logica que implementei cobre corretamente os casos quando o número é divisível por 3, 5 e 7, bem como se existem os dígitos 3, 5 e 7 nos números.

Os testes unitários neste programa abrangem uma grande quantidade de casos, desde cenários mais básicos como numero que não dividem por 3, 5 ou 7, até situações mais complexas como combinações de divisibilidade e múltiplos dígitos dos selecionados. Conta ainda com 2 testes extremos, que não só são divisíveis por múltiplos números 3, 5 e 7 como contem os dígitos 3, 5 e 7.

Os resultados dos testes confirmam que o método funciona corretamente em todas as condições avaliadas. Concluindo assim que a solução que apresentei demonstra boas praticas de desenvolvimento de software, desenvolvendo um código simples, testado e funcional, que cumpre todos os requisitos especificados no exercício fornecido.