Programação Modular: Exercício de Revisão 2

Aluno: Matheus Nolasco Miranda Soares

1) Escolha três fatores de qualidades de software, internos ou externos, de sua preferência. Para cada um, descreva uma situação real(vivida por você), ou hipotética(apenas para exemplificação) na qual houve uma violação do fator de qualidade de software escolhido.

- Robustez: Na empresa em que trabalho há um projeto que aparentava funcionar em todas as situações, porém, quando o cliente foi fazer o dowload dele, há conexão com a internet não era forte o suficiente então na hora de rodar o programa algumas funcionalidades estavam quebradas, após isso o projeto teve que ser remontado para que ficasse o mais leve possível, para evitar esse tipo de problema. A violação ocorreu quando os desenvolvedores não deixaram o software o mais leve e otimizado possível, fazendo com que possíveis erros na hora do dowload acontecessem.

- Legibilidade: Nessa mesma empresa, esse mesmo projeto havia sido feito por programadores free lancers, e quando eles saíram do projeto, e os programadores que trabalhavam fixamente na empresa assumiram, foi muito difícil para eles conseguirem identificar o que cada função e variável fazia, por causa disso, grande parte do código teve que ser reescrita. A violação ocorreu quando os desenvolvedores não pensaram em deixar o código legível, comentado e com nomes de variáveis que façam sentido, o que prejudicou os próximos programadores.

-Integridade: Em um dos projetos que participei nessa empresa, tive que programar uma série de jogos que ficariam em um site, para o registro de informações do jogador em cada jogo, como pontuação máxima, dinheiro, skins habilitadas, utilizei o localStorage. Porém, depois de pronto, percebi que alguns jogadores com mais conhecimento estavam entrando e modificando sua pontuação e liberando skins, através da alteração das variáveis salvas dessa forma, então tive que mudar isso, e passei as variáveis para serem salvas no back-end. A violação ocorreu quando eu não projetei o software para ser o mais seguro possível contra modificações não autorizadas.

2) Explique, utilizando um parágrafo para cada, a influência dos pilares da modularidade na construção de sistemas flexíveis e com baixo custo de manutenção: coesão, encapsulamento, abstração, acoplamento.

A influência da coesão nisso é que, utilizando do conceito da coesão, o desenvolvedor consegue economizar muito na hora de escrever o código, visto que, classes filhas herdam as funções das classes pais, logo, não há necessidade de se reescrevê-las o que deixa o código bem mais flexível e reduz a chance de cometer erros durante as fases de desenvolvimento.

A influência do encapsulamento nisso é que, utilizando do conceito do encapsulamento, o desenvolvedor consegue deixar o sistema mais seguro o possível, fazendo com que futuramente não aconteça problemas de violação de

segurança, e já facilita a manutenção, visto que, isso não será um problema futuramente.

A influência da Abstração nisso é que, utilizando do conceito da abstração, o desenvolvedor consegue deixar o sistema o mais semelhante possível com o que ele representa no mundo real, isso faz com que, em uma futura leitura e/ou manutenção do código, seja mais fácil de identificar o que cada elemento faz dentro do objeto.

A influência do Acoplamento nisso é que, utilizando do conceito do acoplamento, o desenvolvedor consegue eliminar códigos desnecessários, como if e switchs, assim deixando o código mais lindo e fácil de realizar manutenções.

3) Um forno de micro-ondas apresenta as características:

•Pode ser programado com um temporizador, indicando por quantos minutos e segundos ele deve funcionar.

•Pode ser pausado, reiniciado e desligado a qualquer momento pelo usuário.

•Só pode ser ligado se a porta estiver fechada, e sua porta só pode ser aberta quando ele estiver desligado.

•Deve atualizar corretamenteo temporizador quando estiver em funcionamento e sinalizarem a passagem de tempo.

Proponha e escreva os casos de teste unitários que guiem o desenvolvimento da classe Microondas seguindo a metodologia TDD.

```
public class MicroondasTest {
   static Microondas m;
   @BeforeEach
   public void init (){
      m = new Microondas();
   }
   @Test
   public void testarTemporizadorSegundos (){
      this.init();
      m.setTemporizador(0,30);
     assertEquals(30, m.getSegundos());
   }
   @Test
   public void testarTemporizadorMinutos (){
```

```
    this.init();
    m.setTemporizador(5,0);
   assertEquals(5, m.getMinutos());
}
@Test
public void testarAbrirPorta (){
    this.init();
    m.abrirPorta();
   assertEquals("Aberta", m.statusPorta());
}
@Test
public void testarFecharPorta (){
    this.init();
    m.fecharPorta();
   assertEquals("Fechada", m.statusPorta());
}
@Test
public void testarLigarPortaAberta (){
    this.init();
    m.init();
    m.fecharPorta();
    m.Ligar();
   assertEquals("Ligado", m.getStatusMicroondas())
 }
@Test
public void testarLigarPortaFechada (){
    this.init();
    m.init();
    m.abrirPorta();
    m.Ligar();
```

```
   assertEquals("Desligado", m.getStatusMicroondas());
}
@Test
public void testarPausarPassagemTempo (){
    this.init();
    m.setTemporizador(2, 30);
    assertEquals(2, m.getMinutos());
    m.fecharPorta();
    m.Ligar();
    m.Pausar(1, 30);
    m.rodarTemporizador();
   assertEquals(1, m.getMinutos());
   assertEquals(30, m.getSegundos());
}
@Test
public void testarDespausarPassagemTempo (){
    this.init();
    m.setTemporizador(2, 30);
    m.fecharPorta();
    m.Ligar();
    m.Pausar(1, 30);
    m.rodarTemporizador();
    assertEquals(1, m.getMinutos());
    assertEquals(30, m.getSegundos());
    m.Despausar();
    m.rodarTemporizador();
    assertEquals(0, m.getMinutos());
    assertEquals(0, m.getSegundos());
}
@Test
```

```
public void testarReiniciarPassagemTempo (){
    this.init();
    m.setTemporizador(2, 30);
    m.fecharPorta();
    m.Ligar();
    m.Pausar(1, 30);
    m.rodarTemporizador();
    m.Despausar();
    m.Reiniciar();
    assertEquals(2, m.getMinutos());
    assertEquals(30, m.getSegundos());
    m.rodarTemporizador();
}
@Test
public void testarDesligarPassagemTempo (){
    this.init();
    m.setTemporizador(2, 30);
    m.fecharPorta();
    m.Ligar();
    m.Desligar();
    assertEquals("Desligado", m.getStatusMicroondas());
    assertEquals(0, m.getMinutos());
    assertEquals(0, m.getSegundos());
    m.rodarTemporizador();
    assertEquals("Desligado", m.getStatusMicroondas());
    assertEquals(0, m.getMinutos());
    assertEquals(0, m.getSegundos());
}
```

4) Observe as características de uma conta de energia descritas a seguir, de acordo com as regras da distribuidora do nosso estado:

• Uma conta de energia está relacionada a um imóvel, o qual pode ser comercial ou residencial, e tem um número de registro. As contas devem armazenar a leitura do mês anterior e a leitura do mês atual do relógio de luz. O consumo, medido em KWh, é dado pela diferença entre duas leituras consecutivas.

• A tarifa básica é R$ 0,6443 por KWh. Para consumidores de baixa renda, a tarifaé de R$0,5263.

• Há uma taxa extra de R$14,20 para cada 100kWh consumidos.

• O valor total da conta é dado pela soma do valor do consumo com o imposto cobrado sobre este consumo. Para consumidores residenciais, o imposto é de 42,85%. Para comerciais, 33,3%.

• Se o consumo de um consumidor residencial de baixa renda for abaixo de 90KWh, há isenção do imposto e da taxa extra.

Uma mesma pessoa pode possuir diversos imóveis, sendo o cadastro validado por CPF. Tendo em vista ainda as seguintes perguntas importantes

:• Qual foi o consumo de energia de um imóvel no último mês?

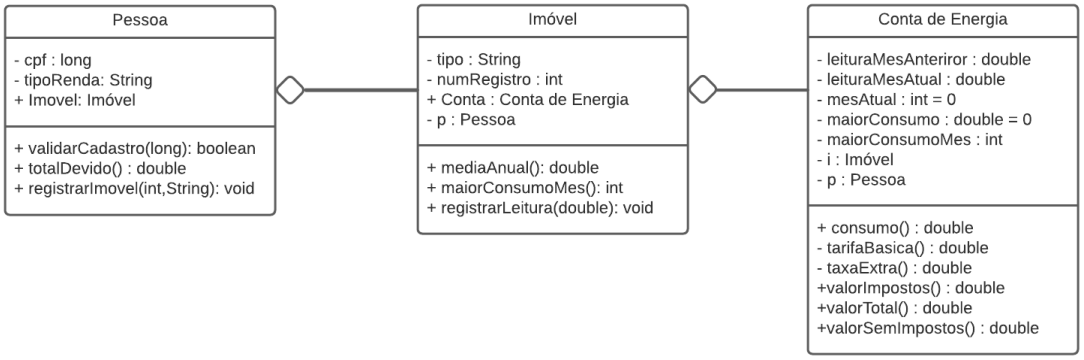
• Qual é o valor total da conta atual de um imóvel?

• Qual é o valor da minha conta sem impostos e o total de impostos pagos?
• Qual é o consumo médio mensal do imóvel no período de um ano fiscal?
• Em qual mês de um ano fiscal houve maior consumo e conta de um imóvel?

• Qual o total devido por um proprietário de imóvel no mês corrente?

a) Modele, usando UML, as classes necessárias para resolver a situação proposta. Não é necessário incluir métodos get/set ou construtores no seu modelo.

b) Além do modelo UML, justifique como cada pergunta acima será respondida utilizando seu modelo.



- O consumo de energia de um imóvel no último mês será respondida através da função consumo(), do objeto “Conta de Energia”, que irá subtrair os valores dos

elementos leituraMesAnterior e leituraMesAtual, os valores desses elementos são passados pela função registrarLeitura(), que chama um construtor de Conta de Energia passando esses valores;

- O consumo de energia de um imóvel no último mês será respondida através da função valorImposto(), do objeto “Conta de Energia”, que utilizará as funções tarifaBasica() , taxaExtra() e o valor dos elementos leituraMesAtual e leituraMesAnterior subtraídos, tais valores serão passados pela função registraLeitura(), da classe Imóvel, que chamará um construtor de Conta de Energia;

- O valor total da conta sem impostos e o total de impostos pagos será respondido pelas funções valorImpostos() e valorSemImpostos(), que, respectivamente, retornarão o valor dos impostos pagos e quanto a pessoa pagaria caso não existissem tais impostos.

- O consumo médio mensal do imóvel no período de um ano fiscal será respondido pela função mediaAnual() da função Imóvel, que calculará, baseado no elemento leituraMesAtual, do objeto Conta de Energia, qual foi o consumo médio por mês em um ano;

- O mês de um ano fiscal que houve maior consumo e conta de um imóvel será respondido pela função maiorConsumoMes(), que utilizará do elemento maiorConsumoMes do objeto “Conta de Energia” para saber qual mês teve maior consumo. Tal elemento será atualizado sempre que um mês superar o último mês que teve maior consumo de energia.

- O total devido por um proprietário de imóvel no mês corrente será respondido pela função totalDevido() do objeto “Pessoa”, que retornará quanto ele precisa pagar por cada um de seus imóveis;