André e Vinícius

Nota: 9.5/10.0

Comentários:  
 - Faltaram alguns modificadores de acesso

- Crie pelo menos duas classes de controle.

- Assinale no diagrama onde há Polimorfismo, Reescrita e Sobreescrita.

- Faltou a suite de teste - Organize as classes em pacotes como os trabalhados em aula.

- Sugiro parametrizar Sla.prioridadeMedia()

- Sempre depois de implementar, remover os TODOs

- Crie duas classes de controle com lógica de programação. Não precisa testar getters e setters.

- Na próxima apresentação, me mostre os testes rodando.

- Temos que incluir nos testes várias condições que possam dar erros (nulos, inválidos, etc.), além dos cenários de sucesso. Lembrem-se que teste não é só verificar se funciona, mas também garantir que é resistente a falhas (teste as exceções, problemas).

Vocês devem arrumar todos estes ítens para a próxima entrega. Vou verificar item a item.

**2ª PARTE**

André e Vinícius

Comentários:

**- Faltaram alguns modificadores de acesso** —> Ainda falta no ControleDoContratoTest ,ImpressoraTest e demais testes

**- Crie pelo menos duas classes de controle**. —> Parcialmente, mas essas classes deveriam ter algo, principalmente a de controle de impressão

**- Assinale no diagrama onde há Polimorfismo, Reescrita e Sobreescrita**. —> Não foi feito

**- Faltou a suite de teste** —> Não foi feito

- Organize as classes em pacotes como os trabalhados em aula. —> Ok

- Sugiro parametrizar Sla.prioridadeMedia() —> Ok

**- Sempre depois de implementar, remover os TODOs** —> Não foi feito

**- Crie duas classes de controle com lógica de programação. Não precisa testar getters e setters**. —> Parcialmente

- Na próxima apresentação, me mostre os testes rodando. —> Ok

**- Temos que incluir nos testes várias condições que possam dar erros (nulos, inválidos, etc.), além dos cenários de sucesso. Lembrem-se que teste não é só verificar se funciona, mas também garantir que é resistente a falhas (teste as exceções, problemas).**—> Não foi feito

**Entrega 2:**

**→-** Um método que não faz nada não tem serventia : ControleDeImpressao.imprimir(Impressora impressora, Integer QtdPaginas). Remova ou implemente.

**→-** - O método ControleDoContrato.adicionaContrato() faz o que um construtor deveria fazer. Uma classe de controle deveria implementar regras de negócio, condições, algo que controla entidades para algum fim. A construção de um objeto deve ser feita pelo próprio construtor do objeto. O método setFranquiaPaginas() está manipulando um objeto que tem tudo para dar erro de nullpointer. Você deve tratar isso e o teste deveria ter pego essa falha.

**→-** - Em geral, os métodos de controle estão bem pobres e a utilidade deles é questionável. É fundamental que achem um papel importante para eles e descrevam no propósito da classe.

**→-** - O método abstrato adicionaCliente faz a mesma coisa que deveria fazer um construtor. Parece que o papel do construtor não está claro para vocês ainda. O adicionar cliente pressupõe colocar clientes em uma lista e não construir uma instância de cliente. Também não faz sentido ter uma lista de cliente em uma classe cliente (a não ser que clientes fossem clientes de outros clientes).

OK- Existe um objeto Date para manipulação de datas, então não use uma String para armazenar essas datas (classe Contrato).

**→-** - Crie um método estático na pessoa física e na pessoa jurídica para validar e formatar o CPF/CNPJ. Esse método parece mais interessante do que o para buscar um contador. A lógica já está implementada em um método de objeto (formatação), pegue-a para validar o CPF com os dígitos verificadores (tem na internet) e retornar uma String formatada.

OK- Faltou a suite de testes novamente!

**→-** - Nenhum teste dos controles foram feitos, isso vai comprometer muito a nota se não implementarem. Lembrem-se que a avaliação da implementação será pelos testes.

**→-** - Tudo que eu tinha falado de testar condições de erro não foram feitas (valores nulos, inválidos, incompletos, etc). Os testes no geral estão muito fracos. Vocês vão tirar nota melhor se implementarem melhor os testes e menos coisas no código do que apenas o código.

- Não foi implementada a interface da linguagem.

- Faltou indicar no diagrama onde as coisas estão (sobrecarga, rescrita, herança, etc… cada um dos itens do trabalho).

Olha, tem muita coisa para fazer e peço que novamente olhem com detalhes os meus comentários e implementem. A maioria foi negligenciada.

**Entrega 3**

**PENDÊNCIAS:**

**→-** - Crie pelo menos duas classes de controle. —> Parcialmente, mas essas classes deveriam ter algo, principalmente a de controle de impressão que está muito pobre —> Continua igual

**→-** - Assinale no diagrama onde há Polimorfismo, Reescrita e Sobreescrita. —> Não foi feito DE NOVO

OK - Faltou a suite de teste —> Não foi feito para as próprias suites —> DE NOVO

**→-** - Temos que incluir nos testes várias condições que possam dar erros (nulos, inválidos, etc.), além dos cenários de sucesso. Lembrem-se que teste não é só verificar se funciona, mas também garantir que é resistente a falhas (teste as exceções, problemas). —> Não foi feito DE NOVO

**→-** - Crie duas classes de controle com lógica de programação. Não precisa testar getters e setters. —> Parcialmente, está bem pobre o papel da classe ControleDeImpressao —> Não foi feito DE NOVO

**→-** - Em geral, os métodos de controle estão bem pobres e a utilidade deles é questionável. É fundamental que achem um papel importante para eles e descrevam no propósito da classe. —> Não foi feito DE NOVO

**→-** - O método abstrato adicionaCliente faz a mesma coisa que deveria fazer um construtor. Parece que o papel do construtor não está claro para vocês ainda. O adicionar cliente pressupõe colocar clientes em uma lista e não construir uma instância de cliente. Também não faz sentido ter uma lista de cliente em uma classe cliente (a não ser que clientes fossem clientes de outros clientes). —> Não foi feito DE NOVO

**→-** - Existe um objeto Date para manipulação de datas, então não use uma String para armazenar essas datas (classe Contrato). Ninguém está usando esses métodos de get e setDate.

**→-** - Crie um método estático na pessoa física e na pessoa jurídica para validar e formatar o CPF/CNPJ. Esse método parece mais interessante do que o para buscar um contador. A lógica já está implementada em um método de objeto (formatação), pegue-a para validar o CPF com os dígitos verificadores (tem na internet) e retornar uma String formatada.

OK - Faltou a suite de testes novamente!

**→-** - Nenhum teste dos controles foram feitos, isso vai comprometer muito a nota se não implementarem. Lembrem-se que a avaliação da implementação será pelos testes.

**→-** - Tudo que eu tinha falado de testar condições de erro não foram feitas (valores nulos, inválidos, incompletos, etc). Os testes no geral estão muito fracos. Vocês vão tirar nota melhor se implementarem melhor os testes e menos coisas no código do que apenas o código.

**→-** - Não foi implementada a interface da linguagem.

**→-** - Faltou indicar no diagrama onde as coisas estão (sobrecarga, rescrita, herança, etc… cada um dos itens do trabalho).

OK - Utilize o padrão de nomenclatura de pacotes apresentado na última aula.

**→-** - Um método adicionaXXX pressupõe que está adicionando um objeto a uma coleção, no caso de Contrato.adicionaImpressora() deveria estar usando Contrato.setImpressora() pois está atualizando um atributo. Inclusive, por que a atributo impressoras está no plural? O mesmo se aplica para adicionaCliente dessa mesma classe (se está atualizando um atributo e não adicionando um objeto a uma coleção).

**→-** - Os métodos adicionaCliente da PessoaFisica e PessoaJuridica está fazendo o papel de um construtor, pois se está criando um objeto com seus valores padrões (atributos de classe). Você deve usar construtor para isso e remover o método abstrato.

**→-** - Rever em todo o sistema o uso de adicionaXXX conforme havia comentado e rever o papel de construtores no sistema (acontece também em impressora adicionando insumo).

**→-** - Insumo poderia ter um construtor padrão que recebe tipo e tempoVidaUtil.

**→-** - O método PessoaFisica.getCPFFormatado() poderia ser estático recebendo como argumento um CPF string e (1) validando e (2) retornando o mesmo formatado. O mesmo se aplica para CNPJ. Procure na internet código pronto para isso (não se esqueça de copiar a fonte no seu código - não tem problema copiar, desde que referencie de onde tirou)

**→-** - Fazia parte do trabalho ter implementado uma interface da linguagem e não foi feito

**→-** - Falta documentação Java em algumas classes, escrever o autor e comentar sobre a funcionalidade da classe.

**→-** - Retirar importações sem uso.

**→-** - Erro de compilação na última asserção do getCNPJTest/getCNPJFormatadoTest e getCPFTest/getCPFFormatadoTest

**→-** - Não possui teste nulo em nenhum caso, apenas teste vazio. Isso é importante pois estamos trabalhando com objetos.

**→-** - Poderia melhorar a organização do diagrama, fica confuso achar as anotações das devidas classes.

OK - O AllTest do pacote teste/controle não está funcionando em virtude de testes não implementados. Quando não implementou, troque @Test por @Ignore para não deixar de testar o que está funcionando.

**→** - ControleDoContratoTest nao está passando em todos os testes —> Erro de nullPointer.

**→** - ContratoTest não está passando em todos os testes.

OK - Crie uma suite de teste para rodar os testes de controle e entidade juntos.

**→** - Não tem teste do ControleDeImpressao

**→** - Os testes em geral estão muito fracos quando implementados.

**→** - getters e setores NÃO devem ser testados! Somente métodos que tenham lógica de programação.

OK - Classe Controle de impressão existe um comentário documentado vazio.

**→** - Classe PessoaJuridica e Contrato não possuem a documentação. Faltou explicar o que acontece nas classes.

**→** - Ainda falta colocar o autor em algumas classes.

**→** - A descrição das classes deveria estar mais padronizada, assim como o autor. Em algumas classes um está acima do outro, em outras o contrário, ou há um @autor para cada integrante da dupla, ou um pros dois.

OK - Faltou o package com o teste de todos os testes.

**→** - Faltaram implementações na classe ControleDeImpressaoTest.

**→** - Classe ControleDeContratoTest deveria possuir teste nulo.

**→** - ContratoTest está muito pobre de testes. Implementar os testes que faltam.

**→** - ImpressoraTest deveria ter mais testes, usando outros tipos de Assert e comparaveis.

**→** - Implementar a parte de null no método void TotalPagar em ControleDeContrato. Essa classe poderia ter bem mais testes.

**→** - assertEquals do testPagar está com valor diferente do esperado na mensagem, e retorna NullPointerException.

OK - AllTests do package entidade não está com todas as classes test do seu pacote (falta teste de Contrato).

**→** - Falta teste null em PessoaJuridicaTest, PessoaFisicaTest e ImpressoraTest.

**→** - public String getNumcontrato() em Classe Contrato precisa de implementação para caso nulo.

**→** - Não possui Parsing.

Da nova entrega:

**→** - Não implementou a classe concreta com herança.

**→** - Não indicou no diagrama onde tem parsing, formatação e manipulação de strings, apesar de estar no código

**NOTA 3.0, entregaram a mesma versão do trabalho anterior e não fizeram o que era solicitado. (lembrando que as entregas incluem a correção da revisão + itens solicitados)**