

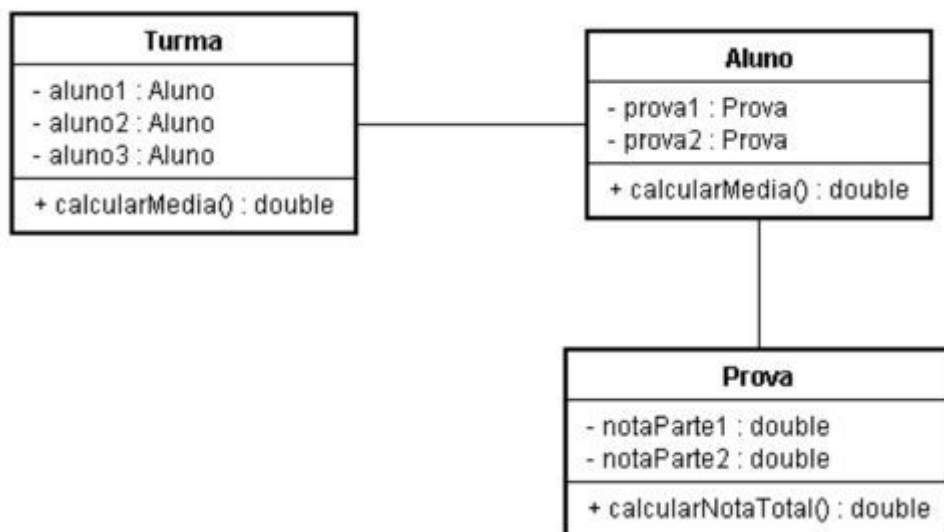
**Universidade Federal do Rio Grande do Norte**  
**Instituto Metr pole Digital**  
**IMD0040 - Linguagem de Programac o 2**  
**Aula07 - Lista de exerc cios**

- Esta lista de exerc cio   composta por 3 quest es. Sendo uma f cil, uma m dia e uma dif cil.
- As quest es valem, respectivamente, 20%, 30% e 50%.
- N o se assustem com o tamanho dos textos.

**Quest o F cil (20%) – Sistema Escolar**

**A tarefa**

Voc  faz parte de uma equipe de desenvolvimento que foi contratada para fazer um sistema para as escolas da rede particular de Natal, entre in meras funcionalidades do sistema voc  ficou respons vel pelo sistema que calcula a m dia das turmas baseado em provas semestrais, com isso onde   composto pelas seguintes classes:



Observe uma descrição sobre o que cada método de cada classe faz:

Classe	Método	Descrição
Turma	calcularMedia()	Calcula a média da turma. A média é calculada utilizando a média de cada aluno da turma.
Aluno	calcularMedia()	Calcula a média do aluno. A média é calculada utilizando a nota total das duas provas realizadas por ele.
Prova	calcularNotaTotal()	Calcula a nota total da prova. Esta nota é dada pela soma das notas das partes 1 e 2. A nota total não pode ultrapassar 10.0.

Crie uma aplicação que instancia uma turma, três alunos na turma e as duas provas para cada aluno. Defina também notas para as provas. A aplicação deve mostrar mensagens informando a média de cada aluno e a média geral da turma.

Para a definição das notas, utilize as seguintes informações:

<b>Aluno 1</b>	Prova 1	Nota Parte 1	4.0
		Nota Parte 2	2.5
	Prova 2	Nota Parte 1	1.0
		Nota Parte 2	7.0
<b>Aluno 2</b>	Prova 1	Nota Parte 1	6.5
		Nota Parte 2	3.5
	Prova 2	Nota Parte 1	0.0
		Nota Parte 2	3.0
<b>Aluno 3</b>	Prova 1	Nota Parte 1	5.0
		Nota Parte 2	4.0
	Prova 2	Nota Parte 1	6.0
		Nota Parte 2	1.5

#### Questão Média (30%) – Projeto world of zuul

Se ainda não tiver baixado o projeto que está no SIGAA disponibilizado na aula passada, baixe. Na classe Game temos um método chamado processCommand, leia o método veja o que ele realiza quando um comando é reconhecido. Visto que uma sequência de instruções if é adicionada cada vez que precisamos adicionar um novo comando de reconhecimento, temos que esse design não é muito bom.

A partir disso, aprimore este design. Projete as classes de modo que o tratamento de comandos seja mais modular e novos comandos possam ser adicionados mais facilmente. Implemente o código e teste.

### Questão Difícil (50%) - A Senha

Baixe o arquivo jogo-senha.zip disponibilizado no SIGAA. Este arquivo contém uma implementação para o jogo senha.

O jogo inicia quando o desafiante seleciona quatro peças coloridas para ser sua senha. Esta não pode ser visualizada pelo desafiante. A senha pode ser formada por qualquer combinação de pinos coloridos mas sem repetir nenhuma cor. O outro jogador tem até 10 tentativas para acertar a senha. Cada tentativa é feita colocando uma fila de pinos coloridos no tabuleiro. Cada fila colocada deve permanecer na mesma posição durante todo o jogo. Depois de cada tentativa, o desafiante responde ao desafiado se ele está no caminho certo, colocando os pinos brancos e pretos. No caso, o computador irá representar o desafiante, ou seja, ele deve gerar automaticamente uma senha com diferentes cores de pinos e colocar os pinos brancos e pretos para os acertos do desafiado.

O pino branco indica que o jogador acertou a cor, mas não acertou a posição do pino da senha. O pino preto indica que o jogador acertou a cor e a posição do pino da senha. Nenhum pino indica que o jogador não acertou nem a cor nem a posição do pino.

Não existe ordem para a colocação dos pinos brancos e pretos. O desafiante não vai informar nem a cor nem a posição correspondente ao pino preto, branco ou espaço vazio.

Exemplo: Caso o desafiado informe 4 cores corretas, mas as 4 cores estiverem em posições incorretas, o desafiante deverá apresentar 4 pinos brancos.

Exemplo 2: Caso o desafiado informe 2 cores corretas com suas respectivas posições corretas e uma cor correta, porém com sua posição incorreta, o desafiante deverá apresentar 2 pinos pretos e 1 pino branco.

A partir disso, edite os códigos disponibilizados do jogo. O mesmo se encontra incompleto e não é possível sua execução. Sabendo disso implemente uma classe chamada Jogo, onde conterá as seguintes funções:

**public boolean verSeAdivinhoGanhouJogo()** - Função que verifica se o Adivinho já ganhou o jogo, sabendo que o mesmo ocorre apenas quando o FornecedorDaSenha retorna todos os pinos pretos.

**public void mostrarPinosTentativaDaJogada()** - Método de impressão da jogada atual efetuada pelo adivinho(jogador atual).

**public void mostrarPinosRetornoDaJogada()** - Método de impressão resultante da jogada do adivinho, apresentando os pinos pretos e brancos indicando se ele acertou uma posição e cor -ou- uma cor de algum pino da senha.

**public void terminarJogo()** - Função que finaliza o jogo, caso o adivinho tenha acertado a sequência de cores, ou caso tenham ultrapassado 10 tentativas.

**public void executarJogo()** - Função que inicializa o jogo Senha, apresenta um menu com as opções do jogo e o progresso jogada a jogada.

Crie os atributos necessários para o bom funcionamento da classe jogo, lembrando sempre de manter a coesão e a coerência entre as classes envolvidas.

Além da classe Jogo, crie a classe Main para executar o jogo Senha.