Antes de começar a resolver as questões, leia atenciosamente as instruções a seguir:
1 - Os exercícios devem ser realizados **INDIVIDUALMENTE**.
2 - As resoluções dos exercícios devem estar em **ARQUIVOS DIFERENTES e uma pasta .zip com todos os arquivos deverá ser entregue até às 23:59 do dia 08/05/2022 pelo Google Classroom.**
3 - Nas descrições das questões, todos os atributos, métodos, classes e/ou interfaces em negrito devem ser explicitamente criados com o **MESMO NOME** nas soluções. Do contrário, você é livre para escolher qual a assinatura dos mesmos.
4 - Os tipos de entrada e saída explicitados devem ser respeitados.
5 - É permitido criar atributos, métodos, classes e/ou qualquer estrutura auxiliar que você considerar necessária para a resolução do problema. Porém, tudo que for pedido deverá **obrigatoriamente** ser implementado.
6 - Os métodos que forem sobrescritos devem ter a anotação **@Override**.
7 - **A correção da lista será feita através da análise individual de cada código, o principal aspecto não se baseia no *output* correto, mas em uma arquitetura de solução condizente com os princípios da programação orientada a objeto. Logo, utilizem *getters*, *setters*, modificações de visibilidade e todos os demais conceitos estudados em sala de aula na medida que julgarem necessário para resolução do problema.**
8 **-** Qualquer dúvida ou inconsistência em relação às questões, deve-se informar imediatamente aos monitores.
9 - Boa sorte!

[Q1] (1,0) Implemente uma classe **Text**, a qual representa um texto qualquer com algumas operações básicas, onde o texto é tido como uma sequência de caracteres alfanuméricos separados por espaço. A classe deverá ter um método **getSize()**, o qual retorna o número total de palavras no texto, **findFreq(String substr)**, que deverá retornar a frequência de uma determinada substring dentro do texto (desconsiderando ser maiúscula ou minúscula, ou seja, EXEmpLo == exemplo), **replaceWords(String str)**, que deverá substituir todas as ocorrências de uma palavra no texto por uma outra palavra qualquer.

Além disso, crie a classe principal **Main**, a qual deverá, em ordem, instanciar um objeto com um texto qualquer, utilizar os dois primeiros métodos citados acima, modificar o texto, utilizar o último método e printar o texto resultante.

[Q2] (1,0) Implemente uma classe **Account** e uma classe **Password**. Uma Account deverá possuir um identificador alfanumérico de no máximo 10 caracteres (String), uma agência (String), um saldo (float) e uma senha (Password), a qual representa uma composição da classe Password na classe Account. A classe Password deverá ter duas senhas (ambas String) definidas para cada instância: uma alfanumérica (formada por letras e números) e uma numérica (formada somente por números). O construtor da classe Account deverá obrigatoriamente receber todos os parâmetros referentes aos atributos. No entanto, a classe Password terá dois tipos de construtores, um com argumentos referente a todos os atributos da classe e outro vazio. No segundo caso, deverá ser definido senhas *defaults*: 123abc para a senha alfanumérica e 123 para a numérica.

Crie a classe principal **Main** que deverá instanciar dois objetos da classe Account, cada um utilizando um tipo de construtor diferente da classe Password. Em seguida, mostre as senhas das duas contas, modifique a senha alfanumérica da primeira conta e a numérica da segunda e mostre novamente as duas senhas, bem como o identificador e a agência da primeira conta. Por fim, modifique o número de identificador e a agência da segunda conta.

[Q3] (2,0) A partir da questão anterior, modifique a classe Account de forma que dois novos métodos sejam criados: **debit(float value)**, o qual debita um determinado valor do saldo e **credit(float value)**, o qual credita determinado valor do saldo. Ademais, crie uma classe **SavingAccount** que herda a classe Account e representa uma conta poupança, a qual deverá possuir um comportamento para render juros, ou seja, considerando uma taxa, deverá ser creditado na conta o valor do saldo multiplicado pela taxa. Para a implementação do comportamento, considere o polimorfismo de sobrecarga, onde a taxa pode ser passada como parâmetro ou não. No segundo caso, deve-se considerar o valor de 3%. Por fim, crie uma nova aplicação da classe **Main** que instancia um objeto do tipo **SavingAccount** e utiliza os três novos métodos criados.

[Q4] (3,0) Rafael é dono de um zoológico cujos funcionários são robôs. O zoológico possui dois tipos de animais: leões e cobras. Ambos devem ser representados por classes que herdam a classe abstrata **Animals**, a qual possui o nome do animal (String), a posição do animal no zoológico e um método implementável **talk()**, que retorna uma string caracterizando o som realizado pelo animal.

Implemente uma classe **Robot** referente ao funcionário robô, o qual irá possuir um identificador alfanumérico de no máximo 5 caracteres (String) e a posição do robô no zoológico.

Por fim, implemente uma classe **Zoo** que deverá conter uma matriz 4x3 (representando o zoológico), um vetor de animais (Animals[]), um vetor de funcionários robôs (Robot[]), a quantidade de animais (int) e a quantidade de robôs (int). A classe deverá ter um método **insert(Animals animal)** para inserir um animal no vetor de animais e um método **insertRobot(Robot robot)** para inserir um robô no vetor de robôs, com ambos retornando um booleano indicando se a inserção foi bem sucedida. Por fim, implemente um método **checkNeighbor(String id, String name)**, que checa se o robô com identificador referente ao primeiro parâmetro e o animal com o nome referente ao segundo parâmetro estão um do lado do outro (considerando as direções norte, sul, leste e oeste). Se sim, o animal deverá falar e a posição do robô deverá ser setada como a do animal, enquanto que a do animal deverá ser setada como a do robô.

Na classe **Main**, instancie um objeto do tipo Zoo, insira um leão e uma cobra, bem como dois funcionários. Verifique se o primeiro funcionário e o leão estão um do lado do outro, o mesmo para o segundo funcionário e a cobra.

Observações: considere que o vetor de animais e de robôs terá um tamanho fixo de 12 e não se esqueça de verificar se a posição de inserção para ambos os casos é válida, ou seja, se não está ocupada (não possui nenhum animal ou robô alocado naquela posição) e se está dentro dos limites da matriz. Você é livre para implementar a solução da maneira que achar melhor, mas não se esqueça de utilizar os conceitos de encapsulamento e abstração.

[Q5] (3,0) Você deverá implementar um um avaliador de expressões. Para isso, crie uma interface **Expression** que possui dois métodos implementáveis: **solve()** e **show()**. O

primeiro deverá retornar o valor resultante de uma expressão (analisada sempre da direita para a esquerda) que poderá ser aritmética, lógica ou ternária. O segundo deverá retornar uma string representando a resolução da expressão.

Implemente as classes **ArithmeticExpr**, **LogicalExpr** e **TernaryExpr** para o tratamento relacionado a cada uma das expressões. A classe **Main** deverá receber uma entrada em formato de string e instanciar o objeto referente ao tipo de expressão, chamando o método solve() e em seguida o método show(). No caso das expressões aritméticas, a resolução deverá considerar a seguinte prioridade de operações: divisão (/), multiplicação (*), subtração (-) e soma (+). No caso das expressões lógicas e ternárias, deverão ser considerados os seguintes operadores lógicos: maior (>), menor (<), maior ou igual (>=), menor ou igual (<=), igual (==).

Observações: considere que os valores dados de entrada sempre serão inteiros. Ademais, você é livre para decidir quais os atributos que deverão ser criados, mas não se esqueça de previamente estruturar a arquitetura necessária para resolução do problema, considerando os conceitos de encapsulamento e abstração.

Exemplos:

- Para uma entrada **2*3+4/6**, o método solve() deverá retornar **6.66666666667** e o método show() deverá retornar **(((2)*(3))+((4)/(6)))**.
- Para uma entrada **2+5/10<12+16** o método solve() deverá retornar **True** e o método show() deverá retornar **(((2)+((5)/(10)))<((12)+(16)))**.
- Para uma entrada **2*3+4/6!=3/4?3<5+6:26*7*6/4+3** o método solve() deverá retornar **True** e o método show() deverá retornar **(((2)*(3))+((4)/(6)))!=((3)/(4)))?((3)<((5)+(6))):((((26)*(7))*((6)/(4)))+(3))**.