Linguagem de Programação II

https://www.w3schools.com/java/java_classes.asp

Ementa

> ~~Recursividade~~.
> ~~Classes abstratas e associações~~.
> ~~Interfaces~~.
> ~~Tratamento de exceção~~.
> Integração com banco de dados.
> Reflexão.
> ~~Serialização.~~
> ~~Enumeração~~.
> Coleções.
> Programação em camadas.
> Empacotamento de aplicações

Aula 2

- [ ] Crie um projeto Java no Netbeans
- [ ] Crie um pacote nesse projeto chamado br.com.interfaces
- [ ] Crie uma interface **Automovel** com pelo menos 2 métodos
- [ ] Crie uma classe **Carro** e uma classe **Moto** que implementam a interface **Automovel**
- [ ] Crie um método na classe **Carro** que seja exclusivo dela
- [ ] Crie uma classe de **Testes**, contendo o método *main* e faça a chamada dos métodos em objetos das classes **Carro** e **Moto**
- [ ] Altere a implementação de **Automovel** para torná-la uma classe abstrata e coloque pelo menos um de seus métodos como abstrato.

Aula 3

- [ ] Crie uma hierarquia de classes para representar os diferentes tipos de **funcionários** de um escritório que tem os seguintes cargos: **gerente**, **assistente**, **vendedor**. Escreva uma classe base abstrata chamada **Funcionario** que declara um método abstrato *calculaSalario*, que não recebe parâmetros e retorna um valor do tipo *double.*

  - Um gerente recebe duas vezes o *salarioBase;*
  - um assistente recebe o *salarioBase;*
  - um vendedor recebe o *salarioBase* mais uma comissão definida no construtor de sua classe.

- [ ] A classe **Funcionario** deve ainda conter os atributos: *nome* (String), *matricula* (String) e *salarioBase* (double). Use encapsulamento e forneça um construtor que receba os valores correspondentes a serem armazenados nos respectivos atributos.

- [ ] A classe **Funcionario** deverá ser estendida pelas classes representativas dos tipos de funcionários **Gerente**, **Assistente** e **Vendedor**. Em cada uma dessas, sobrescreva o método *calculaSalario* da seguinte forma:

- [ ] Crie uma classe **Teste** com um método *main* que cria um objeto de cada tipo de funcionário. Calcule então a soma dos salários dos três funcionários e imprima no console o valor total.

Aula 4

- [ ] Escreva uma classe abstrata chamada **Mensagem**. Essa classe representa todos os tipos de mensagem e conterá apenas um atributo: *destinatario* (String). Nessa classe você deverá também declarar o método publico e abstrato *mostrar*, que não recebe parâmtros e retorna *void*. Crie as classes filhas da classe **Mensagem**: **Informativo**, **Promocao** e **Urgente**.

- [ ] Cada uma dessas classes deve conter um construtor que receba o *destinatario* e deve implementar o método *mostrar*, exibindo uma mensagem concatenada com o nome do destinatário e que seja específica para o tipo da mensagem.

- [ ] Escreva uma classe **Teste** e no método *main* desta crie um *array* de **Mensagem**. Insira instâncias dos 3 tipos de cartões neste *array*. Após, use um laço *for* para exibir as mensagens.

- [ ] Em que linha(s) acontece polimorfismo nesse código?

Aula 5

- [ ] Crie uma classe **Aluno** que contenha os atributos nome (literal), idade (inteiro) e data da matrícula. Crie um construtor que inicialize os atributos de classe.
- [ ] Crie uma classe de **Testes** e em seu método *main*, crie 2 instâncias de **Aluno**.
- [ ] Para cada **Aluno**, faça com que o usuário insira os dados, usando a entrada de dados via classe **Scanner**. Use os valores digitados pelo usuário para incializar os objetos da classe **Aluno**.
- [ ] Crie um método na classe Aluno que exibe uma mensagem dizendo se ele é maior de idade ou não

Aula 6

https://www.arquivodecodigos.com.br/dicas/3573-java-recursividade-exercicios-resolvidos-7-um-metodo-recursivo-que-calcula-o-numero-de-fibonacci-para-um-dado-indice.html

- [ ] Implemente um método recursivo que receba como entrada um número inteiro positivo N e retorne o seguinte cálculo:

$1 + 2 + 3 + 4 + ... + N$

- [ ] Implemente um método recursivo que receba como entrada um número inteiro positivo N e retorne o seguinte cálculo:

$1 + 1/2 + 1/3 + 1/4 + ... + 1/N$

- [ ] Implemente um método recursivo que receba como entrada dois números inteiros x e k e retorne o valor da operação x^k. Não utilize o método *Math.pow()* do Java. Utilize apenas multiplicação.

Aula 7

- [ ] Crie uma classe abstrata **Matriculado** que tenha como atributos a *matrícula* e um array de *notas* (podem ser fracionárias) e um método abstrato: c*alcularMedia*.

- [ ] Crie uma classe **AlunoIntegrado** que tenha como atributos o nome e o telefone. Essa classe deve estender de **Matriculado** e, em seu construtor, deve inicializar o array *notas* com 3 posições. O método *calculaMedia* deve ser implementado usando a seguinte regra: $media = (0.3 * n1 + 0.3 * n2 + 0.4 * n3)/3$, sendo $n1, n2$ e $n3$ as posições do array *notas*.
- [ ] Crie um método na classe **AlunoIntegrado** para receber do usuário os valores das notas e coloque-as nas posições correspondentes do array *notas*. O usuário deve digitar primeiro qual nota ele quer preencher ($n1, n2$ ou $n3$), indicando o índice do array ($0, 1$ ou 2) e depois digitar o valor.
- [ ] Trate a exceção caso o usuário digite um índice que não exista no array.
- [ ] Lance uma nova exceção caso o usuário digite uma nota negativa.
- [ ] Repita a implementação da classe **AlunoIntegrado** mas agora com uma classe **AlunoSuperior** que contém apenas 2 notas. A regra para calcular a média nessa classe segue a regra: $media = (0.4 * n1 + 0.6 * n2)/2$

Aula 8

- [ ] Crie uma classe que aceite a digitação de dois números e faça a divisão entre eles exibindo seu resultado. Sua classe deve tratar as seguintes exceções:
  - *ArithmeticException* quando ocorrer uma divisão por zero.
  - *InputMismatchException* quando o valor informado não é numerico.
- [ ] Crie uma classe que contenha um *array* de inteiros de 10 posições. Feito isso, permita que o usuário digite valores inteiros a fim de preencher este *array* até que se digite o valor 0. Uma vez digitado o valor 0, o mesmo deve ser inserido no vetor e a digitação de novos elementos deve ser interrompida. Feita toda a coleta dos dados, exiba-os no console. Sua classe deve tratar as seguintes exceções:
  - *ArrayIndexOutOfBoundsException* quando o usuário informar mais que 10 valores.
  - *InputMismatchException* quando o usuário informar um valor que não é numerico.

Aula 9

- [ ] Escreva um programa que armazene os nomes dos meses do ano em um vetor, em seguida peça ao usuário que digite um valor inteiro. Após isso, mostrar o nome do mês correspondente ao número digitado. O programa deve finalizar

quando for digitado o valor zero. Deve-se tratar, através de exceções, a digitação inválida do índice e mês inválido.

- [ ] Construa uma classe **ContaCorrente** , com os atributos: *limite* que armazena a quantidade atual disponível do limite da conta que o usuário possui; o atributo *saldo* que é o valor que realmente é pertencente ao usuário todos *float*.
- [ ] Encapsule os atributos de ContaCorrente.
- [ ] Construa os métodos: *sacar*, *depositar* e *definirLimite*. Todos esses métodos recebem o valor como argumento. Na construção dos três métodos faça com que eles lancem *exceptions* relacionados aos argumentos, por exemplo, sacar, depositar ou definir um limite com valor negativo. Lance também uma *exception* no caso de se tentar sacar um valor maior que o possível.
- [ ] Crie uma superclasse **Conta** e transfira os atributos e métodos pertinentes de **ContaCorrente** para que uma nova classe **ContaPoupanca** possa herdar de **Conta** e também ter acesso a esses atributos e métodos.
- [ ] Crie uma classe **Testes** com o método *main* e crie instâncias das classes **ContaCorrente** e **ContaPoupanca**. Solicite ao usuário que informe valores para depositar, sacar e definir limite. Exiba no console tanto as mensagens disparadas pelas exceções (caso aconteçam) quanto os valores guardados em cada atributo das contas.

Aula 11

- [ ] Crie uma classe **Cliente** com os atributos: *id*, *nome*, *idade*, *telefone*. Faça um programa para solicitar os dados de vários clientes e armazenar em um ArrayList até que se digite um número de ID negativo. Em seguida exiba os dados de todos os clientes no console
- [ ] Crie uma classe **Agenda** que pode armazenar 10 pessoas, seja capaz de realizar as operações abaixo e que use um *array* simples para o armazenamento:
    - void *armazenaPessoa*(String nome, int idade, float altura); // deve verificar se há espaço disponível no array
    - void *removePessoa*(String nome); // deve procurar por nome para verificar se a pessoa foi incluída
    - int *buscaPessoa*(String nome); // informa em que posição da agenda está a pessoa (se existir)

- void *imprimeAgenda*(); // imprime os dados de todas as pessoas da agenda
- void *imprimePessoa*(int index); // imprime os dados da pessoa que está na posição "i" da agenda.

- [ ] Repita o exercício anterior usando um ArrayList para armazenar as pessoas e altere os métodos.