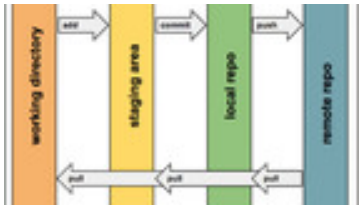
Repositório de POO

---

01. Preparação

Exercícios de fup para aquecer o coração e o cérebro.



@038 Introdução ao git

@000 Estressados A: Busca

@040 Estressados B: Contagem

@032 Estressados C: Tranformações

@039 Estressados D: Controles

02. Uma classe

Nesse módulo, vamos aprender como criar uma classe e como se relacionam atributos e métodos. - No **Carro**, você controla as pessoas que entram e saem, a gasolina e sai para passear. - Na **Calculadora**, você pode realizar operações desde que haja bateria, o resultado pode ser consultado no visor.

@001 Calculadora

@002 Carro

03. Modificadores de Acesso

A partir desse módulo, você deve estar atento aos modificadores de acesso `public` e `private`. Utilize os métodos `set` para controlar as alterações realizadas nos atributos. - No **Tamagotchi**, o setFome(valor) garante que a fome nunca seja maior que o máximo ou menor que 0 e muda o flag para morto caso chegue a 0. - No **Relógio**, o setHora(valor) garante que a hora sempre será um valor entre 0 e 23. - No **Porquinho**, o getValor() só deve retornar o valor se o porco estiver quebrado.

	@036 Relógio
	@006 Tamagotchi
	@017 Porquinho
	@042 Carro Get e Set

04. Agregação Simples

A partir desse módulo, você precisará estar atento às relações entre classes e objetos. Na agregação, a relação entre eles não define vínculos de tempo de vida. Caso o todo seja destruído, o objeto que está sendo agregado não é destruído. - A **Lapiseria** recebe o objeto grafite, enquanto estiver escrevendo, ela vai gastando o grafite e quando o grafite for muito pequeno e não der mais para escrever ela remove o objeto grafite ficando com `null` na referência. A lapiseira não "cria" o grafite, ela apenas recebe o objeto. Também não o destroi, mas devolve sua referência. - Na **Motoca**, a moto recebe e referencia a criança que está a utilizando para passear no parque.

@004 Grafite

@003 Motoca

05. ArrayList I - Arrays

Aqui, começaremos o uso de estruturas de dados como arrays e listas ligadas. Lidaremos com estruturas de tamanho fixo e tamanho variável. - Na **Budega** temos uma fila de espera e um vetor de caixas atendendo. A fila de espera começa vazia e cresce em tamanho conforme os clientes entram e decrementa conforme os clientes são chamados ao caixa. Os caixas entretanto são um vetor de tamanho fixo. Se existem 4 caixas, você modela como um vetor de tamanho 4, onde todos os valores iniciais são `null`. Para saber se um caixa está livre, você deve verificar se o valor do vetor nesse índice é `null`.

@037 Budega

@020 Lapiseira

06. ArrayList II - Lista circular

@041 Tabuleiro

@009 Pula pula

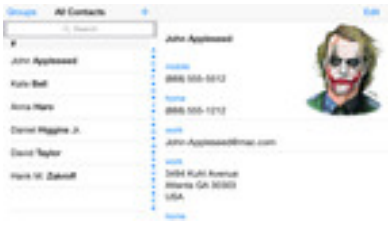
07. ArrayList III - Array de tamanho fixo

@010 Cinema

@012 Topic

@011 JunkFood

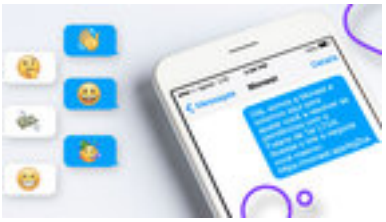
08. ArrayList IV - Organização



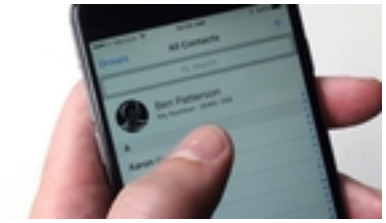
@014 Contato

@007 Tarifas

09. Troca de Dados e Ordenação



@024 Mensagens



@015 Busca

10. Excessões

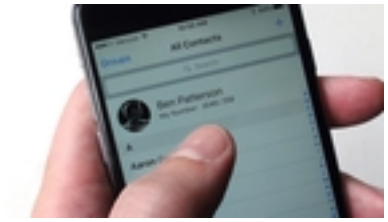
	@008 Agiota
	@028 Bilheteria

11. Mapas Excessões

	@029 Anotações
	@031 Clinica Veterinária

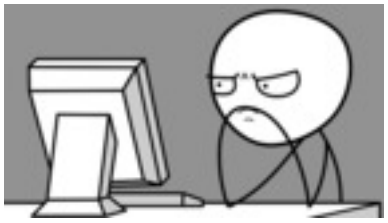
12. Herança I

@013 Cadastro



@030 ContatoStar

13. Lógica I



@021 Matrícula

@025 Twitter

14. Herança II

@023 Salário

@022 Hospital I

15. Lógica II

@026 Whatsapp I

@033 Trem 1

16. Herança III

@034 Trem 2

@027 Whatsapp II - Chat Pessoa