**Anotações sobre Back-end**

- Real = objeto, ideal = ser (inspiração)

- Construtor = espaço para características do ser

- Objeto = instância do construtor

- Main = mundo real (é transitório), Classe = ideal (é eterno)

- Todo objeto tem atributos e ações (métodos)

- Função dentro de uma classe = método

- Variável dentro de uma classe = atributo

- Em Main é variável e função msm

- Construtor = é identificado quando o método tem o mesmo nome da classe

- Classe = embalagem e método = bolo

Diagrama de classes:

- Método tem (), se não é atributo

C#:

- Atributos e classes com letras maiúsculas

MVC:

- Model, View, Controller

- Tudo passa pelo Controller

ASP.NET:

- Mapeia suas classes e cria os controllers automaticamente

- Cria regras de segurança automaticamente

- Objeto = se sobrepõe, é o pai

- encapsulamento = é uma técnica para proteger as informações de uma classe (atributos)

Como? Deixar tudo privado e aplicar get e set onde necessário

- Ponteiro = É uma variável que aponta para um local na memória (outra variável), o que permite manipulação sem ser do objeto pai

- Todo objeto é ponteiro

- Array também é objeto

- Se comparar duas variáveis com == ele compara o endereço na memória

- Máquina virtual = roda o sistema separadamente do hardware, é um simulador

Sobrecarga de métodos ou construtores: Vários métodos/construtores com o mesmo nome, não dá problema se eles tiverem parâmetros diferentes (assinaturas diferentes) e funções diferentes

**Resumo**

**Estrutura básica de uma Classe:**

- Atributos (Encapsulamento (public e private são modificadores de acesso))

+ Construtor (Iniciar atributos)

+ Métodos

+ To String (Para visualização do estado atual)

\*Namespace = pasta virtual para agrupamento de coisas parecidas

**Classes C#**

Exemplo:

Random (gerar números pseudoaleatórios)

- Estáticas (não precisa instanciar pq tem uma instância fixa):

* Console
* Math
* Convert
* Datetime

- Finais (não pode ser herdada ou alterada, toda vez que mudamos o valor, na verdade criamos uma nova variável e apontamos para ela (computador funciona assim também)):

* String

Herança: herdar atributos da classe pai (o dela + pai), por isso em método tem override