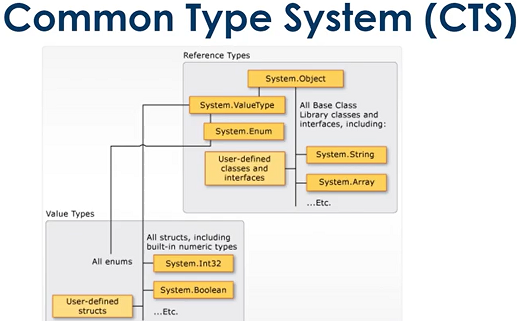
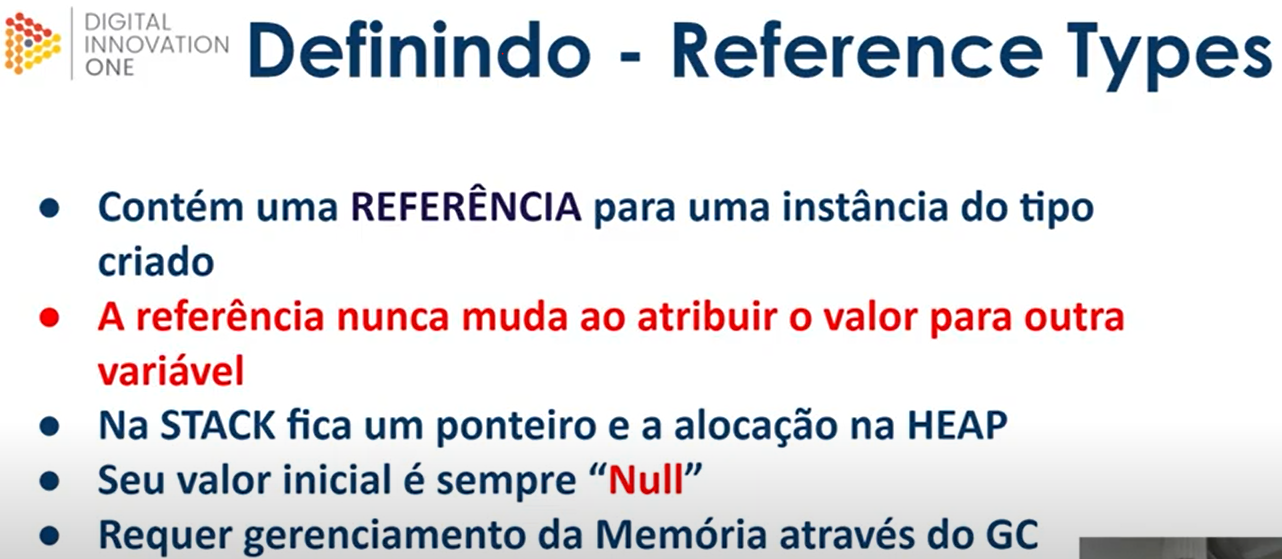
**COMMON TYPE SYSTEM (CTS)** 

**REFERENCES TYPES**

O compilador irá tratar com reference type. Temos todas as classes bases e todas as bibliotecas, interfaces, system string, array, listas, e todo os tipos de estrutura que herdar do system object e tudo que eu converter para object se for necessário trabalhar com object, tudo isso será tratado pelo compilador com references types.



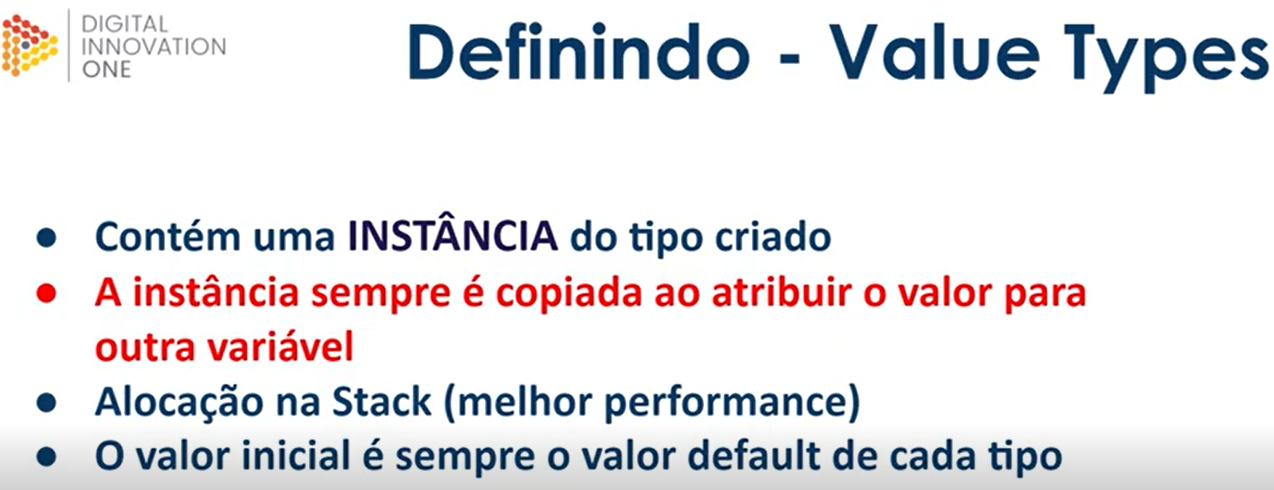
Todo o conteúdo que está na stack temos algo chamado contexto/frame quando o contexto morre, tudo que está dentro dele morre junto, porém na references type o contexto vai morrer e a heap permanece, a heap é uma única coisa durante todo o ciclo de vida da aplicação. Sendo assim eu preciso de alguém que fica esperto que no caso é o CG (garbage collection) que vai limpando essa heap conforme os objetos vão perdendo a utilidade.

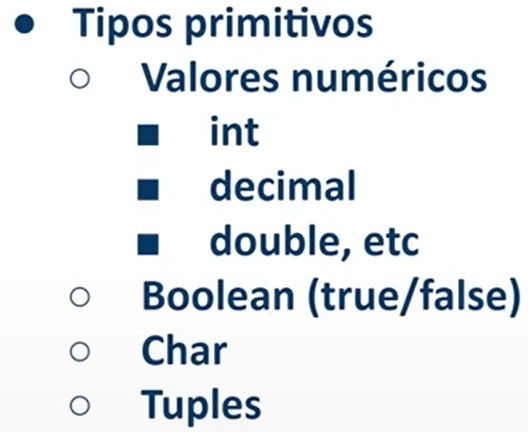


**VALUE TYPES**

Temos os enums, built-in, int e todos os tipos numéricos como decimal e float por exemplo, boolean e todos os structs, tudo isso é tratado pelo compilador como value types.

Importante saber que todos os value types começam com um valor inicial ao ser instanciados, o int por exemplo começa com 0 e o boolean com o default false.



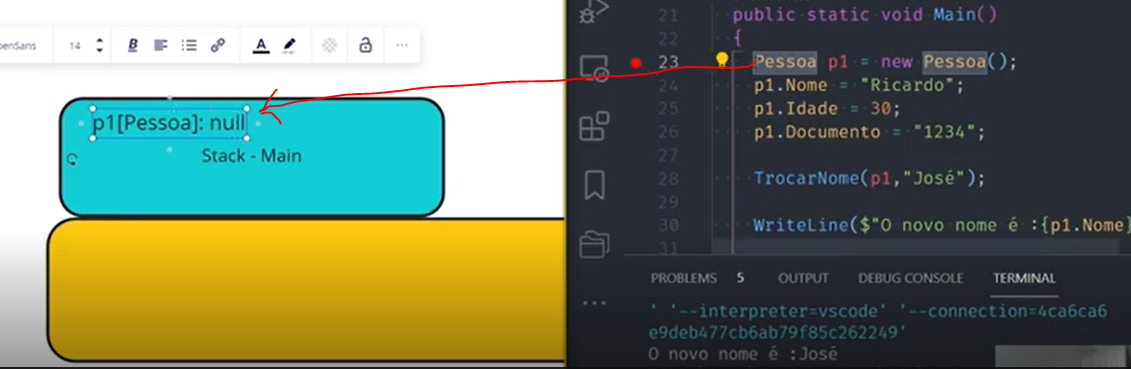


**EXEMPLO DE FUNCIONAMENTO DA HEAP e STACK**

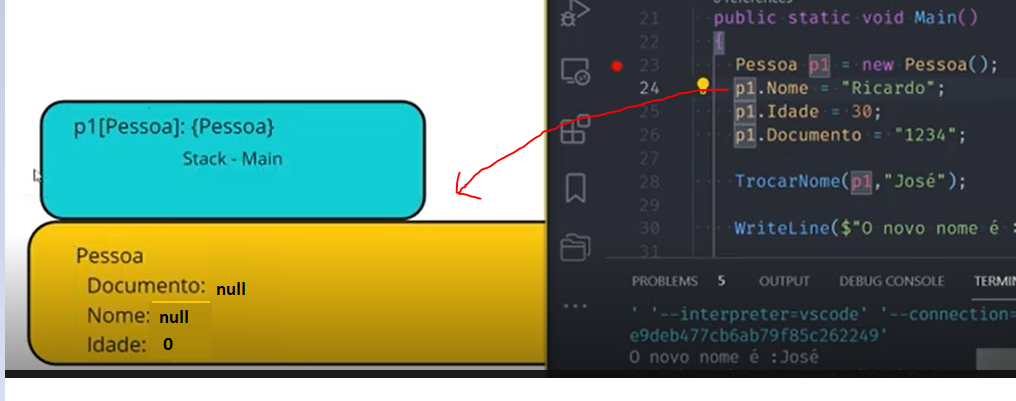
Iniciamos com os dois espaços na memória



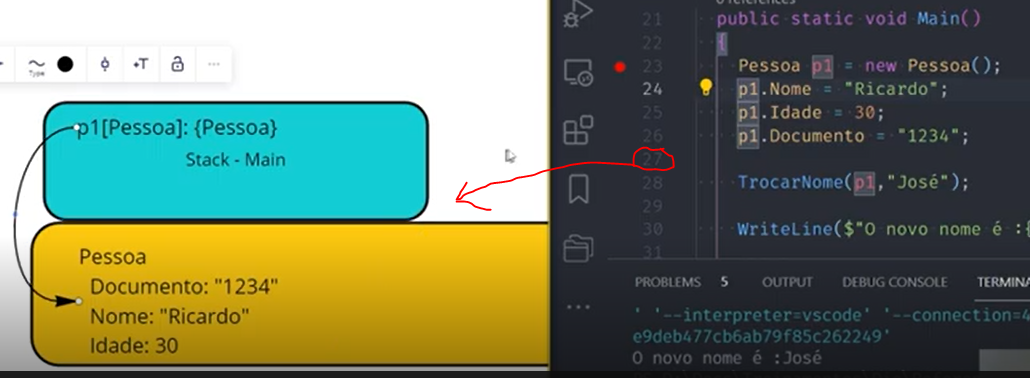
Antes executar a linha que cria o método pessoa ele irá criar na stack do main uma referência null.



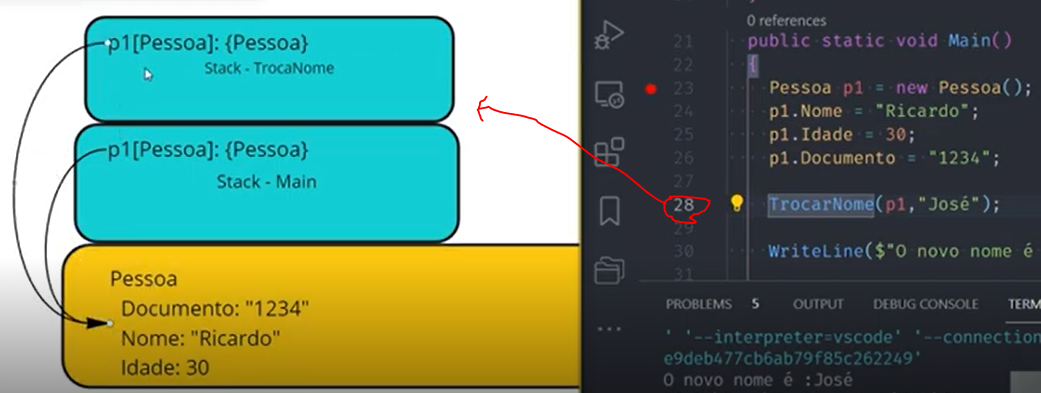
Depois de executado a linha de criação.



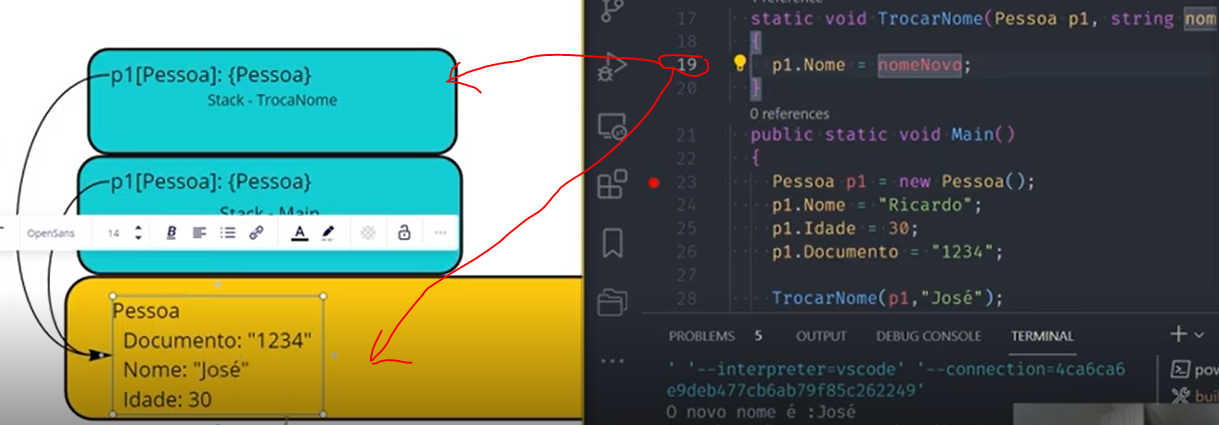
Após passar as linhas de atribuição dos valores irá ficar dessa forma.



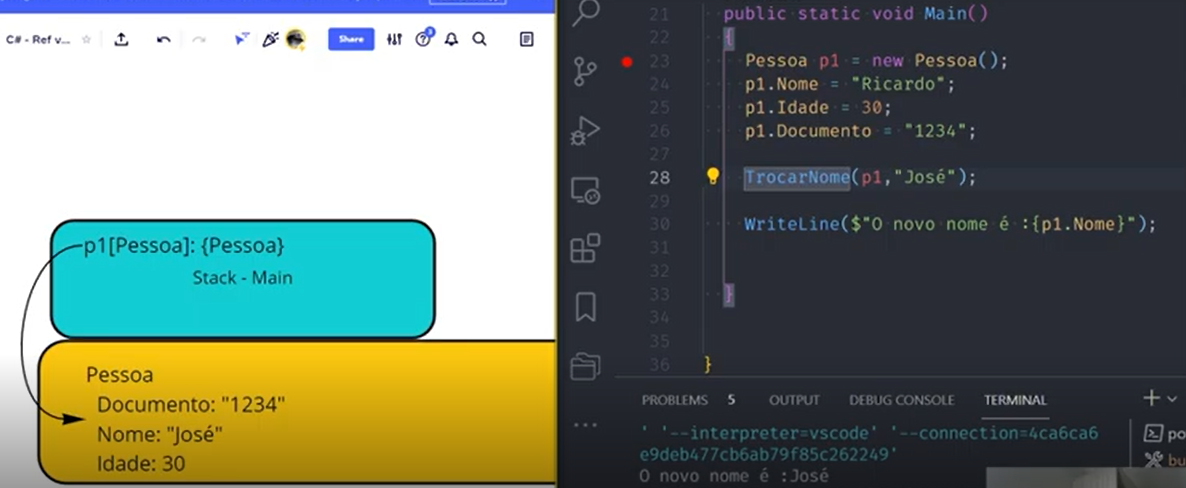
Agora na linha 28 vamos chamar o método que troca o nome.



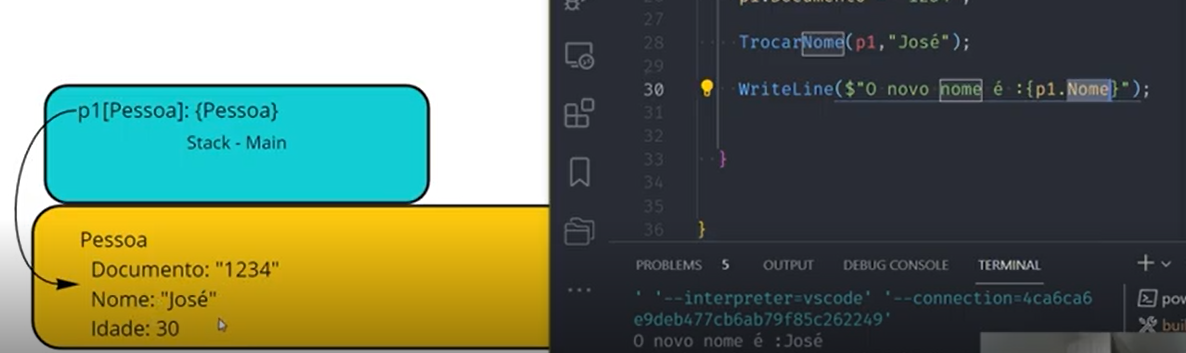
Ao acessar o método trocar nome que passa o nome José no lugar de Ricardo o que acontece é



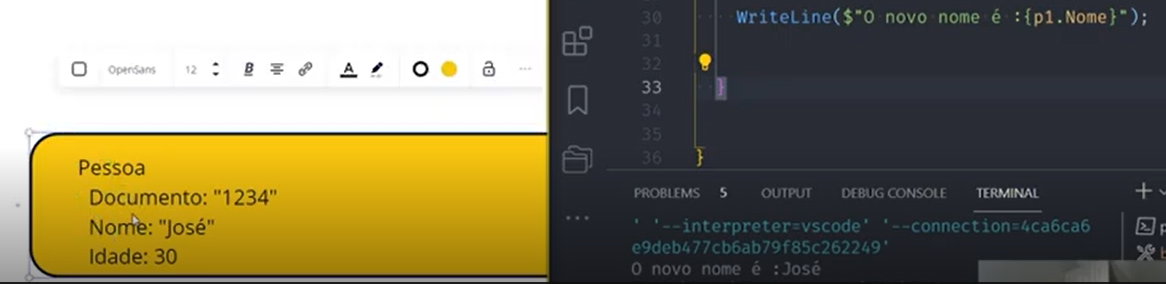
Quando termina de processar o método trocar nome ele irá morrer e com ele tudo que está dentro dele, ou seja, a referência para a heap.



Ao chegar na linha para imprimir na tela ele irá procurar onde está o P1 que tem sua referência mantida na stack da MAIN, porém com os valores já alterado.



Ao chegar na linha 30 que termina a aplicação quem irá morrer é a stack do método main.



O garbage collection se encarregará do que ficou na heap, pois uma vez que não existe mais referência então ele não está mais sendo usado por ninguém.