Aplicando os conceitos de orientação a objeto no JAVA.

Para poder ilustrar a aplicação dos conceitos de O.O dentro da linguagem de programação JAVA iremos utilizar um **pequeno fragmento** do que seria um sistema de caixa eletrônico de um banco (instituição financeira), começaremos com uma única classe e na medida que o curso avançar vamos adicionar mais duas.

Inicialmente vamos trabalhar apenas com o código e ambiente de texto fazendo entradas e saídas em uma espécie de terminal de comando, isso vai ajudar a fixar os conceitos fundamentais de JAVA e de O.O e posteriormente vamos trabalhar com interface gráfica integrada ao código.

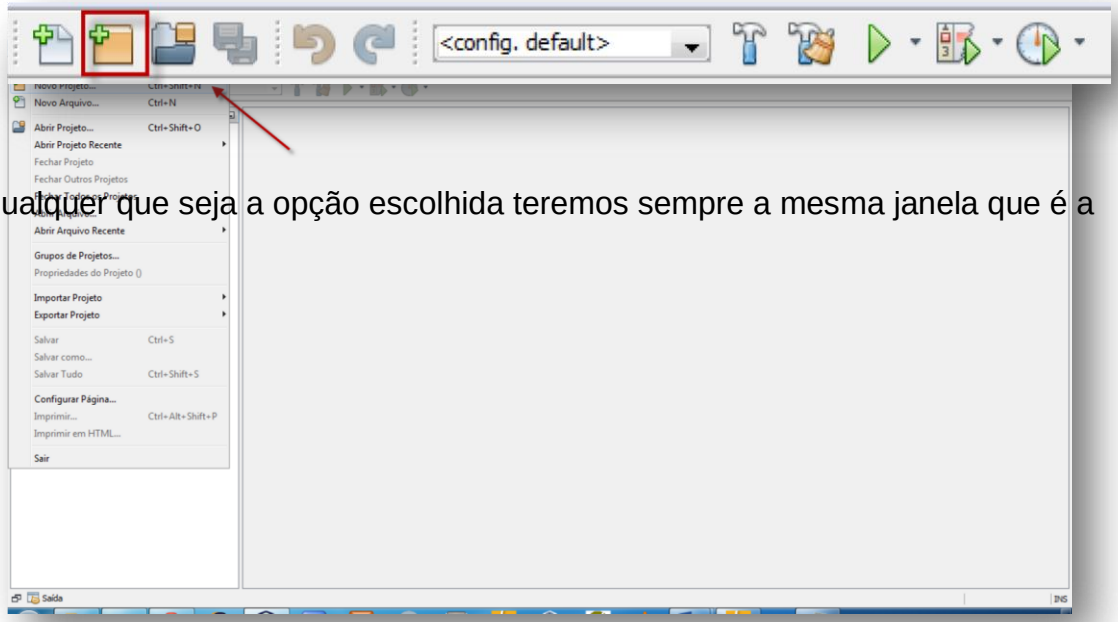
A imagem abaixo representa essa primeira classe através do diagrama de classes da UML (Unified Modeling Language) que é uma linguagem de modelagem de sistemas orientada a objetos muito usada para projetar sistemas baseados na O.O.

Como podemos observar a classe se chama "**Conta**" (primeiro retângulo) e possui três atributos (segundo retângulo) que conforme já foi dito definem as características de uma classe, são eles, "nro**_conta**", "**nro_agencia**" e "**saldo**" inicialmente os dois primeiros não terão utilidade somente saldo vai sofrer ações nessa etapa do projeto. A classe também possui quatro métodos (terceiro retângulo) que definem as ações ou as responsabilidades da classe, são eles, "**saldo**", "**sacar**", "**depositar**" e "**sair**".

Criando um projeto no NetBeans.

Existem várias formas de iniciar um novo projeto no NetBeans, usando as teclas de atalho "**crtl + shift + n**".

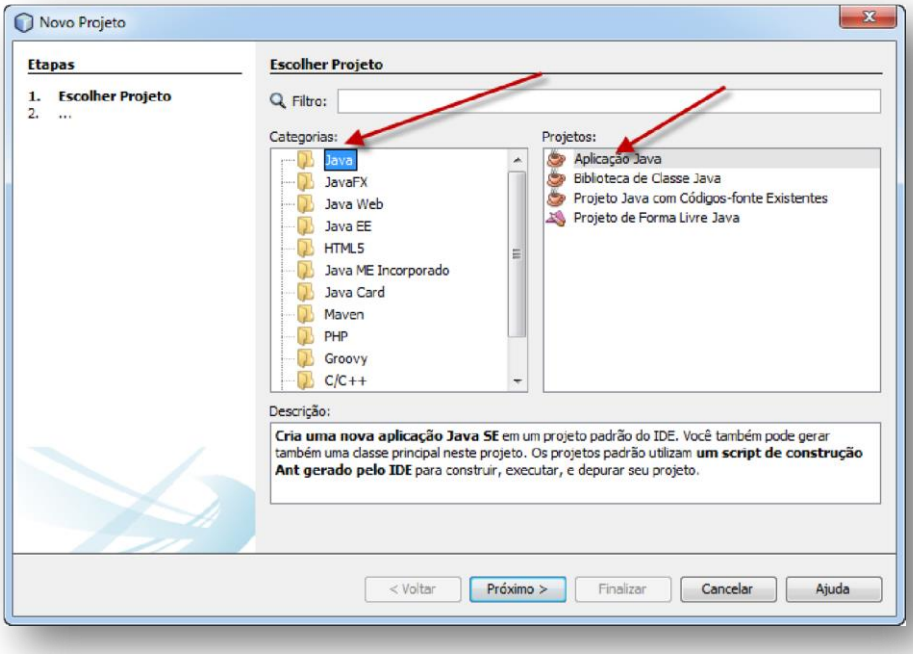
abaixo:
abaixo:



Qualquer que seja a opção escolhida teremos sempre a mesma janela que é a

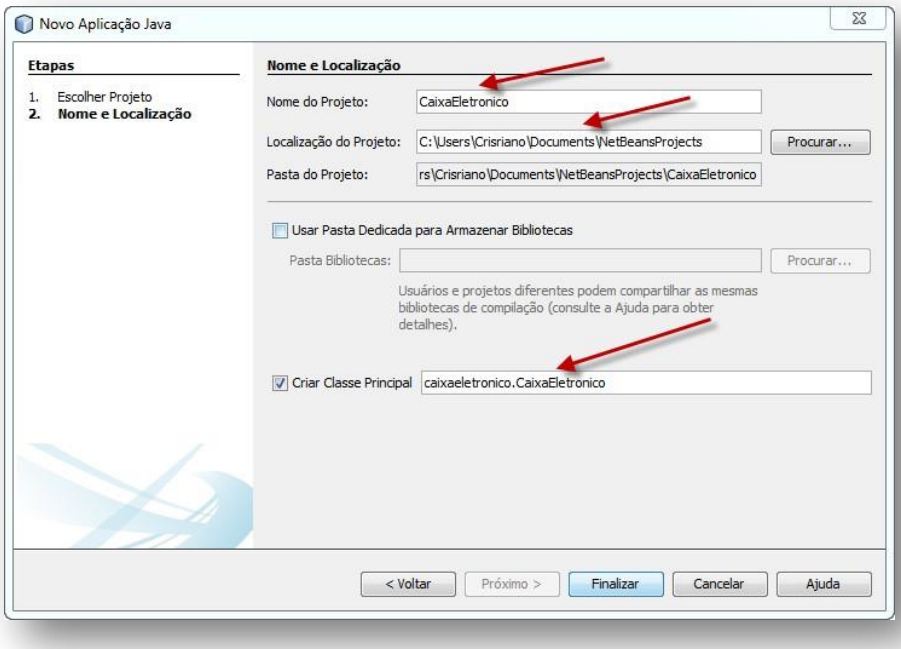
E o botão de novo projeto na barra de ferramentas, conforme ilustra a imagem
Através do menu "**Arquivo > Novo Projeto**", conforme ilustra a imagem

janela de "**Novo Projeto**" onde devemos escolher qual é o tipo de projeto que queremos criar, observe a imagem abaixo:



Como pode ser visto na imagem acima existem várias categorias de projetos em JAVA e em outras linguagens e cada categoria traz um subgrupo que define o tipo do projeto que queremos criar. No nosso caso vamos criar uma categoria de projeto JAVA e o tipo do projeto é "**Aplicação Java**" conforme o destaque das setas.

A janela seguinte solicita dados referentes ao nome do projeto, localização do projeto e criação de uma classe principal, observe a imagem abaixo:



Nesse caso o nome do projeto ficou como “ **CaixaEletronico**” a ausência de acentuação e proposital e altamente recomendada, além disso apenas as primeiras letras de cada palavra são maiúsculas, isso não é obrigatório, mas é uma convenção na comunidade de desenvolvimento JAVA e mais uma vez é altamente recomendado, observe também que não existe espaço entre as palavras e novamente não é obrigatório e novamente é uma convenção da comunidade de desenvolvimento altamente recomendada.

A terceira informação que podemos alterar aqui é a localização do projeto por padrão quando instalamos o NetBeans é criada uma pasta chamada “NetBeansProjects” no caso do Windows essa pasta é criada em documentos, poderíamos mudar a localização se quiséssemos eu recomendo não mudar.

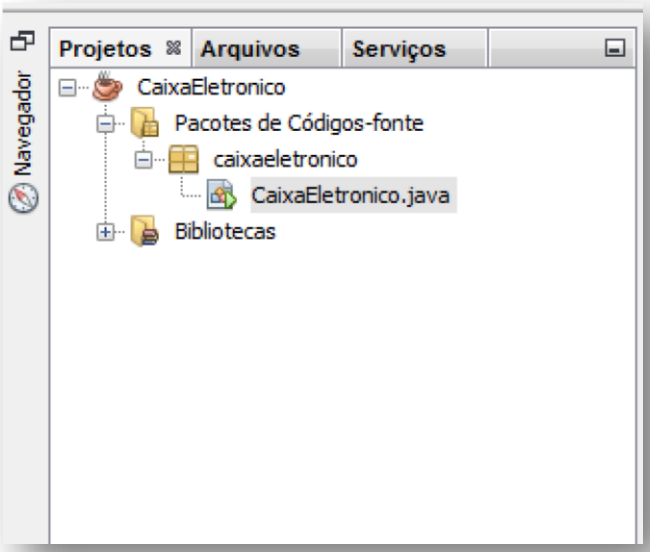
Por último temos a opção de criar uma classe principal junto com o projeto, essa opção já vem marcada por padrão nesse caso é altamente recomendo manter essa configuração. Observe a sintaxe apresentado nessa última opção “**caixaeletronico.CaixaEltronico**” o texto a esquerda do ponto final é o pacote onde a classe principal vai ser criada e o texto a direita é a classe em si.

Pacotes são formas de organizarmos nossas classes dentro de um projeto JAVA é o mesmo conceito das pastas do Windows e altamente recomendado que cada categoria de classe tenha um pacote especifico, isso torna o projeto mais organizado, existem até padrões de projeto para isso um muito conhecido é o **MVC (Model View Control)**.

Uma vez que todas as configurações estejam feitas basta clicar sobre finalizar e o projeto será criado.

A estrutura de um projeto JAVA.

Existe um padrão estrutural hierárquico para projetos JAVA a imagem abaixo ilustra tal estrutura:

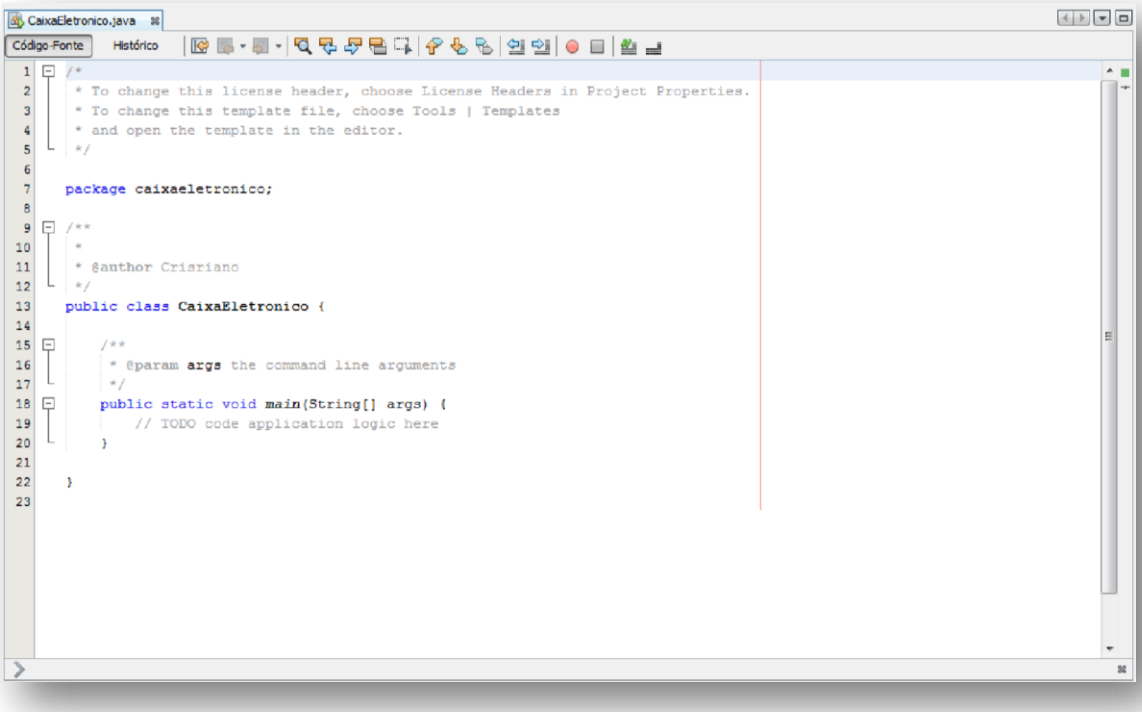


Observe que "**xicara de café**" é a parte da estrutura mais "**alta**" e representa o projeto em si, logo abaixo dessa vem outras duas pastas "**Bibliotecas**" que é onde estão as APIs do JAVA, podemos inclusive adicionar mais APIs e a outras se chama "**Pacotes de Código-fonte**" que como o próprio nome sugere é onde os pacotes e códigos de nosso projeto ficarão alocados, observe que o pacote padrão (cubo caramelo) criado junto com projeto está dentro dessa pasta e dentro do pacote padrão está a nossa classe principal.

A classe principal.

A classe principal é a classe responsável pela execução do projeto em si, quando estamos "rodando" uma aplicação JAVA é a classe principal ou melhor é um objeto da classe principal que está sendo executado é nessa classe que a maioria dos objetos das outras classes do sistema são criados.

Observe a imagem abaixo:



A linha sete (7) define a qual pacote a classe pertence, lembre-se é possível e altamente recomenda ter mais de um pacote nos projetos.

A linha treze (13) declara a classe como publica, ou seja pode ser vista dentro de todo o escopo do pacote a palavra "**class**" define que é uma classe seguido pelo nome da classe, nesse caso "**CaixaEletronico**", observe também que o nome do arquivo e o nome da classe são os mesmo inclusive na grafia, isso é obrigatório, logo após o nome da classe temos um chave "**{**" que indica o início ou abertura do corpo da classe, observe também que temos outra chave fechando "**}**" na linha vinte e dois (22) que indica o fim da classe ou o fechamento do corpo da classe é dentro desse limite que são criados os atributos e os métodos de uma classe.

Da linha dezoito (18) até a linha vinte (20) temos a declaração de um método "**especial**" o método "**main**" esse é o método principal e é ele que torna a classe "**CaixaEletronico**" a classe principal da aplicação.

Quando executamos uma aplicação JAVA essa por sua vez é executada dentro da JVM que interpreta os códigos JAVA e os traduz para o sistema operacional, assim que a aplicação JAVA é executada a JVM procura por uma classe que possua o método "**main**" e começa a construção da aplicação por ali.
As características do método main:

public – define que o método é público, ou seja pode ser manipulado por qualquer objeto da aplicação.

static – esse comando torna o método "**main**" especial, métodos estáticos não precisam ter sua classe instanciada em objetos para serem executados, como a classe principal é a primeira a ser executada e é a responsável por instanciar a maioria dos objetos de uma aplicação JAVA não se cria um objeto dela, daí a necessidade do método "**main**" ser executado sem a criação de um objeto de sua própria classe, podemos criar nossos próprios métodos estáticos.

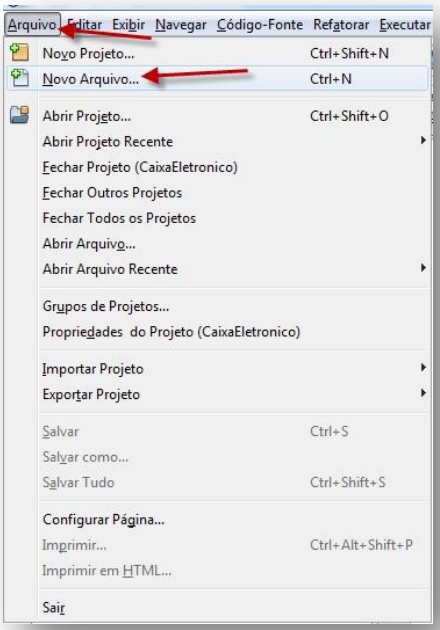
void – Assim como as funções podemos definir valores de retorno para os métodos o comando "**void**" define que o método em questão não retornará nenhum valor após sua execução.

Observe que assim como a classe o método "**main**" também possui uma chave que abre seu corpo uma que o fecha, nesse caso o corpo do método "**main**" ainda está vazio, logo se executarmos a aplicação (**F6**) nada acontecerá.

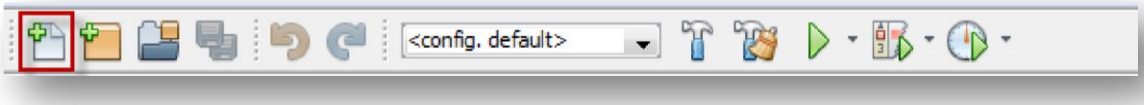
Criando a classe "Conta".

Para criar uma nova classe precisamos criar um novo arquivo e existem várias formas disso ser feito no NetBeans, através das teclas "**ctrl + n**".

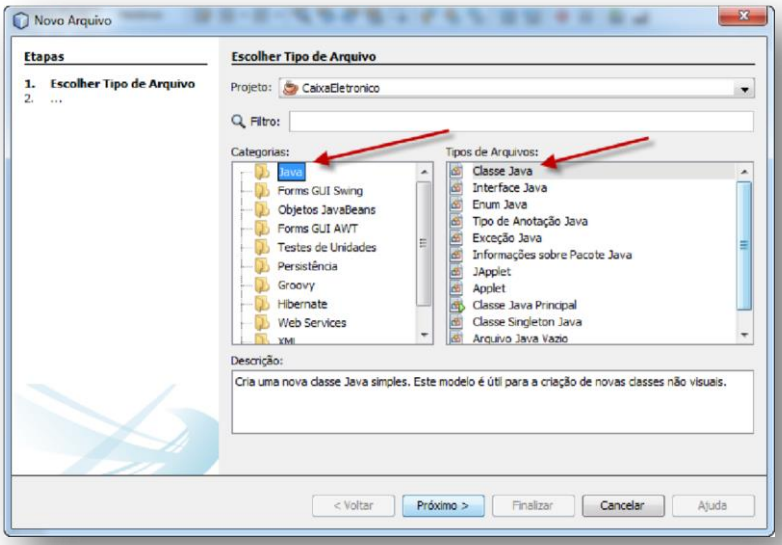
Através do menu "**Arquivo > Novo**", observe a imagem abaixo:



Através da barra de ferramentas, observe a imagem abaixo:

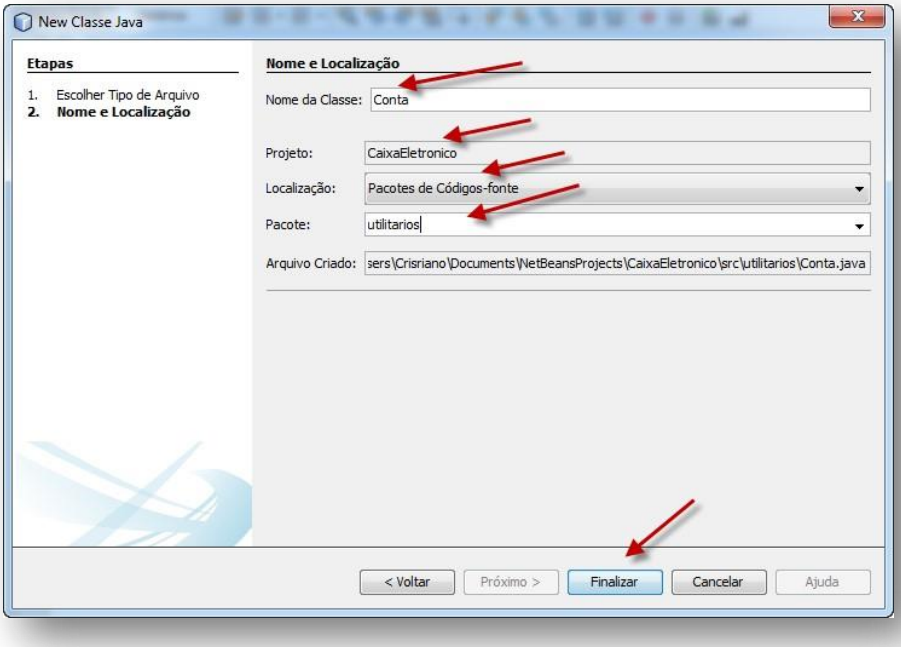


Qualquer que seja a opção escolhida teremos sempre a mesma janela que é a janela de "**Novo Arquivo**" onde devemos escolher qual é o tipo do arquivo que queremos criar, observe a imagem abaixo:



O grupo da esquerda define a categoria e o da direita o tipo de arquivo no nosso vamos usar primeira opção de categoria grupo "**Java**" e dentro desse no grupo da direita o tipo "**Classe Java**".

Basta clicar em próximo e na tela seguinte vamos configurar as opções do arquivo que vai representar a classe que queremos criar, observe a imagem abaixo:



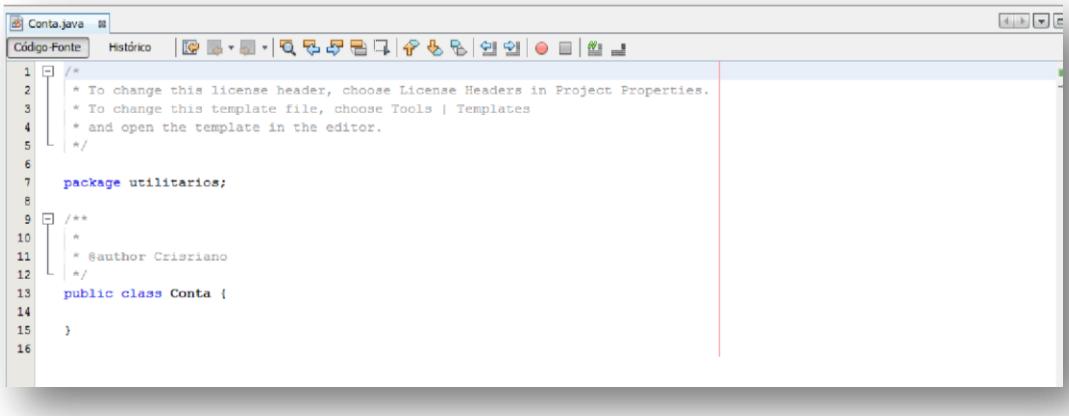
O primeiro item a ser configurado é obvio, nesse caso o nome ficou “ **Conta**”, observe que a primeira letra está em maiúsculo o que não é obrigatório, porém é uma convenção altamente recomendada que inclusive é seguida pelo próprio JAVA.

Em seguida temos o projeto ao qual a classe vai ser inserida e não podemos mudar isso nesse ponto do processo assim como o item seguinte que se refere a localização da classe dentro da estrutura do projeto.

O terceiro item faz referência ao pacote e no meu caso eu adicionei a palavra “**utilitarios**” (**sem o acento**) esse pacote não existe então o NetBenas vai criar

e incluir a classe automaticamente. Agora basta clicar sobre o botão finalizar. note que é diferente do pacote da classe principal vista anteriormente.

A imagem abaixo ilustra a classe "**Conta**" criada:



Observe a linha sete (7) fazendo referência ao pacote que a classe pertence, Da linha treze (13) a linha (15) a classe é declara e como podemos notar seu corpo está vazio.

Tipos de dados no JAVA.

O próximo passo lógico na construção de nossa classe seria a criação e configuração dos atributos, mas antes disso vamos falar dos tipos de dados no Java.

O JAVA é o que se chama de linguagem fortemente "tipada" isso significa que ao criarmos um atributo, variável ou constante no JAVA precisamos atribuir um tipo a qualquer uma dessas.

A tabela abaixo mostra os principais tipos de dados no JAVA:

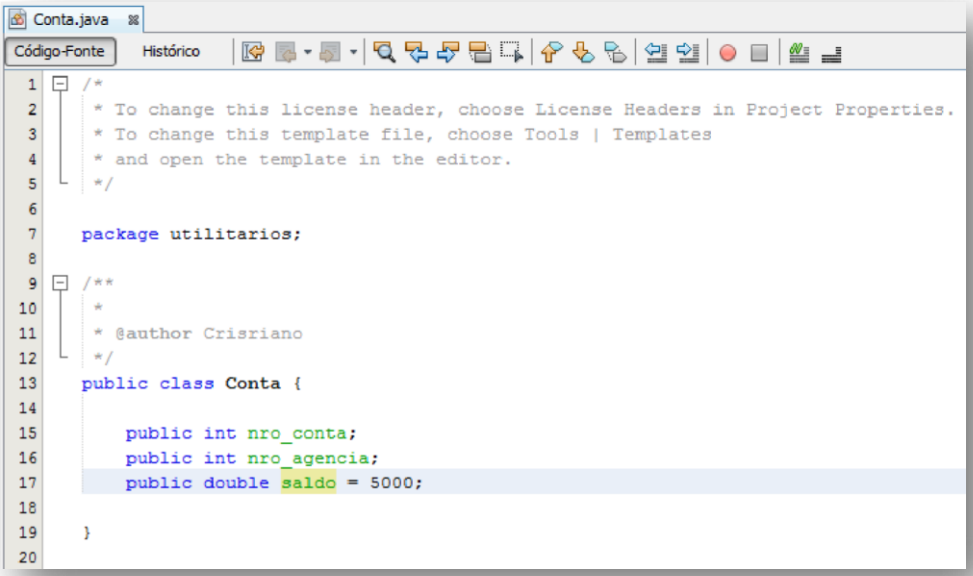
Tipos de dados no JAVA	
Tipo	Descrição
boolean	Pode assumir o valor true ou o valor false
char	Caractere em notação Unicode de 16 bits. Serve para a armazenagem de dados alfanuméricos. Também pode ser usado como um dado inteiro com valores na faixa entre 0 e 65535.
byte	Inteiro de 8 bits em notação de complemento de dois. Pode assumir valores entre -27=-128 e 27-1=127.
short	Inteiro de 16 bits em notação de complemento de dois. Os valores possíveis cobrem a faixa de -2-15=-32.768 a 215-

	1=32.767
int	Inteiro de 32 bits em notação de complemento de dois. Pode assumir valores entre -231=2.147.483.648 e 231-1=2.147.483.647.
long	Inteiro de 64 bits em notação de complemento de dois. Pode assumir valores entre -263 e 263-1.
float	Representa números em notação de ponto flutuante normalizada em precisão simples de 32 bits em conformidade com a norma IEEE 754-1985. O menor valor positivo representável por esse tipo é 1.40239846e-46 e o maior é 3.40282347e+38
double	Representa números em notação de ponto flutuante normalizada em precisão dupla de 64 bits em conformidade com a norma IEEE 754-1985. O menor valor positivo representável é 4.94065645841246544e-324 e o maior é 1.7976931348623157e+308
String	Cadeia de caracteres.

Definido os atributos da classe “Conta”.

Conforme visto anteriormente os atributos definem as características de uma classe, ou seja, são os atributos que “carregam” os dados de uma classe. Devemos declarar sempre os atributos dentro do corpo da classe e fora do corpo dos métodos, isso permite que todos os métodos da classe possam ter acesso aos atributos e consequentemente manipula-los, por convenção se declara atributos sempre no início da classe, mas esse podem ser declarados em qualquer parte dessa, desde que sempre fora dos métodos.

A imagem abaixo ilustra a declaração dos atributos da classe “**Conta**” das linhas quinze (15) a dezessete (17):



O comando "**public**" define a visibilidade da classe falaremos mais sobre isso no tópico de encapsulamento por hora basta saber que a visibilidade "**public**" permite o livre acesso aos atributos, logo após definirmos o nível de visibilidade de um atributo devemos definir seu tipo, lembre-se o JAVA é fortemente "**tipado**", logo não podem existir atributos sem um tipo de dados associado a ele e por fim definimos o nome do atributo.

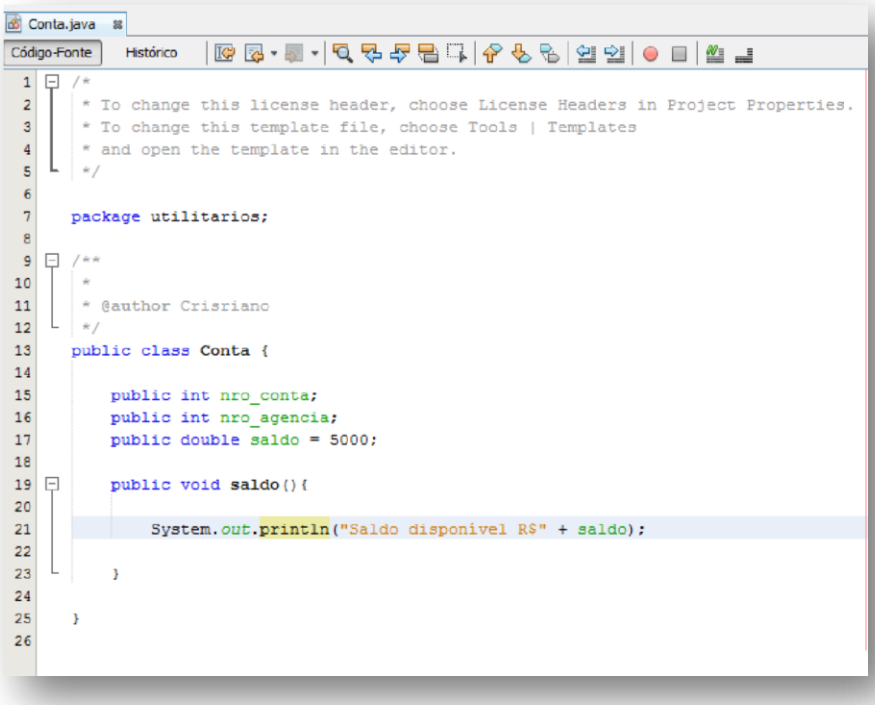
Observe que o atributo "**saldo**" foi configurado com o valor "**5000**" de forma manual isso se faz necessário aqui apenas para testarmos os métodos que criaremos a seguir em uma aplicação real esse valor estaria armazenado em banco de dados, veremos isso também mais à frente.

Observe também quem o NetBeans sinaliza atributos com a cor verde, sempre que encontrar alguma palavra em verde significa que é um atributo.

Criando o primeiro método.

Como já foi dito anteriormente métodos são as estruturas da O.O. responsáveis pelas tarefas (responsabilidades) que uma classe executa, um método bem construído é altamente especializado e podemos descrever sua funcionalidade em uma única sentença sem uso de virgulas, ou seja, um método bem feito realiza uma tarefa especifica, isso permite a reutilização desse código uma das características mais fortes da O.O.

A imagem abaixo ilustra o método saldo:



A linha dezenove (19) inicia a declaração do método como "**public**" sem retorno após sua execução "**void**" nomeado como "**saldo**" e se observarmos existem dois parênteses abrindo e fechando, dentro desse espaço poderíamos definir parâmetros para o método que nada mais são do que dados que um método exige para realizar sua tarefa, veremos isso nos métodos de saque e deposito e por fim nessa mesma linha temos a chave "**{**" de abertura do corpo do método.

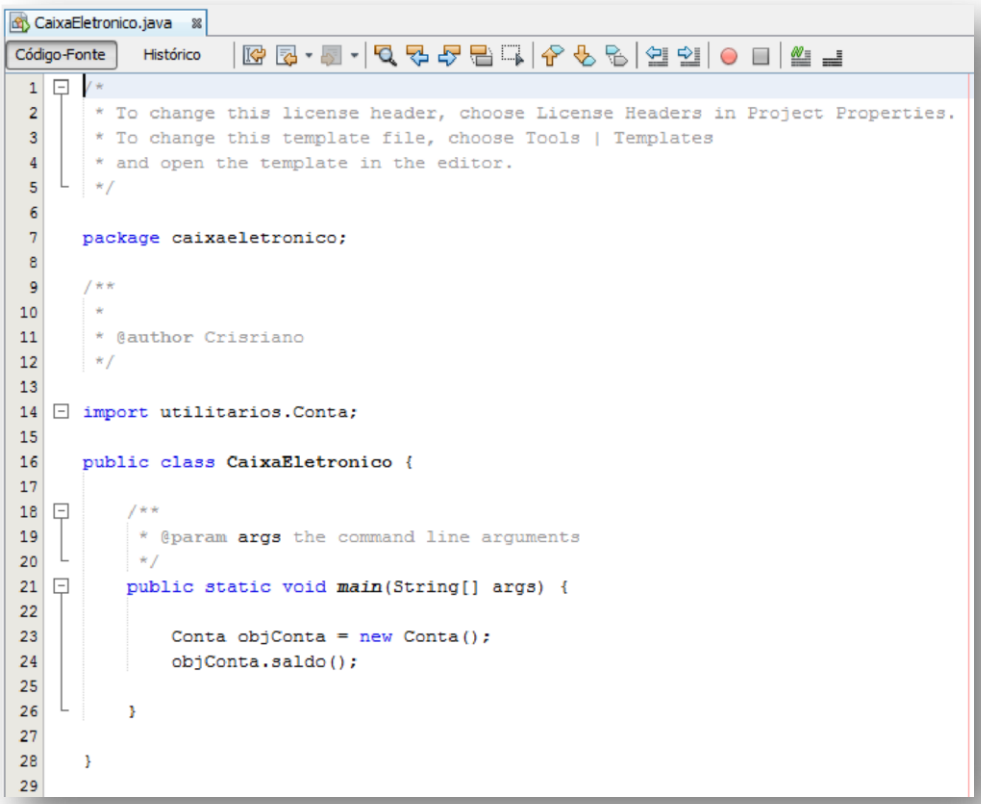
A linha vinte e um exibe o valor do atributo saldo através do comando "**System.out.println**", "**Sytem**" é uma classe do JAVA que implementa várias ações básicas como entrada e saída de dados, "**out**" é um atributo dessa classe que carrega métodos estáticos e "**println**" é um método que gera uma saída "**output**" em modo de texto, existem vários métodos "**print**".

É claro que esse método saldo é apenas um exemplo de aplicação de conceito um método saldo de verdade traria os dados de uma base de dados como veremos mais adiante nesse curso.

Instanciado um objeto da classe "Conta".

Como vimos anteriormente uma classe não pode ser executada diretamente para tal precisamos criar um objeto dessa, lembre-se classe são como "**moldes**" ou "**plantas**", ou seja generalizações de categorias de objeto do mundo real representados dentro de uma solução computacional.

Nossa primeira versão da classe conta vai ter um objeto criado dentro da classe principal "**CaixaEletronico**" para ser mais exato dentro do método "**main**" dessa classe, observe a imagem abaixo:



A linha vinte e três (23) do corpo do método " **main**" cria um objeto da classe "**Conta**", para tal se faz necessário a criação de uma variável de instancia de objeto que nesse caso se chama "**objConta**" o nome poderia ser qualquer coisa válida dentro do contexto de declaração de variáveis.

Observe que a esquerda da variável "**objConta**" temos a palavra "**Conta**" isso porque mesmo na hora de declarar variáveis de instancia de objeto a regra da "**tipagem**" do JAVA continua valendo, logo quando criamos uma variável de instancia de objeto precisamos definir um tipo para essa e o tipo é sempre o nome da classe, ou seja, uma variável de instancia de objeto possui o tipo da classe que representa, isso se chama tipo por referência.

Após definir o tipo e o nome da variável de instancia de objeto devemos criar o objeto em si, observe que a direita da variável "**objConta**" temos um sinal de atribuição "**=**" indicando que algo vai ser configurado na variável, logo em seguida temos o comando "**new**" indicando que um novo objeto vai ser criado e finalmente temos o comando "**Conta()**" informando quem é a classe que vai servir de modelo para o objeto. Os parênteses são necessários todas as vexes que informamos a classe que vai ser usada na criação do objeto o nome que se a esses parentes dentro desse contexto é "**construtor padrão da classe**" no momento oportuno iremos explorar mais esse conceito por hora basta saber que ele deve estar lá sempre.

A linha vinte e quatro (24) realiza uma chamada de método, observe que usamos o nome da variável de ambiente seguindo de um ponto, isso indica ao JAVA que estamos acessando algo do objeto criado, pode ser um atributo ou um método, nesse caso vais ser o método "**saldo()**" que vai informar o valor configurado no atributo "**saldo**".