CAPÍTULO 1
LINGUAGEM JAVA

CAPÍTULO 4 – LINGUAGEM JAVA

Uma vez familiarizados com uma parte do universo das siglas e com os ambientes de desenvolvimento relacionados à linguagem Java, é o momento de aprofundar o conhecimento do tema e das boas práticas de programação, na construção de softwares orientados a objetos. Para isso, nosso foco de estudo será a construção de um projeto. Para começar, é importante entender diversos conceitos que ajudarão a estruturar e a padronizar os programas desenvolvidos com essa técnica. A construção de softwares independentes, de fácil expansão, de alto nível de reusabilidade (a possibilidade de reutilizar trechos de um programa) e que viabilizam a interação entre programas e plataformas diferentes, depende dessa padronização, graças à adoção dos conceitos da orientação a objetos. Deixar de usar esses conceitos não gera, necessariamente, erros de compilação, de lógica ou de execução. Porém, faz com que os softwares gerados não apresentem as vantagens inerentes à orientação a objetos, como, justamente, o aumento da reusabilidade e a facilidade de manutenção e de expansão.

A linguagem de programação Java possui as seguintes características:

- A linguagem é case sensitive;
- O uso de letras maiúsculas e minúsculas nos apresenta características do código (convenções);
- Blocos de código são colocados entre chaves { };
- Ao término de cada instrução utilizamos ponto e vírgula;
- Podemos definir uma instrução em mais de uma linha (texto livre);
- Normalmente colocamos uma classe por arquivo.

4.1. CONVENÇÃO DE NOMES

Outro termo comumente usado é convenção (de nomenclatura). Há alguns anos, cada programador escrevia seus códigos de uma maneira bem particular, criando seus próprios estilos e manias. Isso não atrapalhava a execução dos programas (desde que fosse utilizada uma boa lógica, além de adequados recursos das linguagens). Porém, com o passar dos anos e a evolução dos sistemas, o trabalho em equipe se tornou inevitável e vários programadores (muitas vezes distantes entre si, fi sicamente), passaram a dar manutenção em códigos comuns. A partir daí, surgiu a necessidade de adotar determinados padrões de escrita de códigos. Há algumas convenções de nomenclatura bem antigas e consolidadas, como a notação húngara, amplamente utilizada em C e C++. Mas, na prática, as linguagens e até mesmo as empresas acabam criando suas próprias convenções. Tal como ocorre com os conceitos, deixar de usá-la não gera erros de lógica, de compilação ou de execução. Seguiremos, entretanto, à risca as recomendações de conceitos e convenções, para que sejam conhecidas em profundidade e também em nome da implantação de boas práticas de programação.

- Nome de classes com a primeira letra da(s) palavra(s) maiúscula.

```
public class Conta { ... }
public class ContaCorrente { ... }
public class CaixaEletronico { ... }
```

- Constantes devem ser definidas com todos os caracteres maiúsculos.

```
public class Produto {                    constante
  public static final int COR_BRANCO = 0;
}
```

- Nomes de variáveis e métodos com a primeira letra minúscula

```
public class Conta {                      variável
   private double saldoDaConta;
   public void deposito(double valor) {
     ...                                  método / função / procedure
   }
}
```

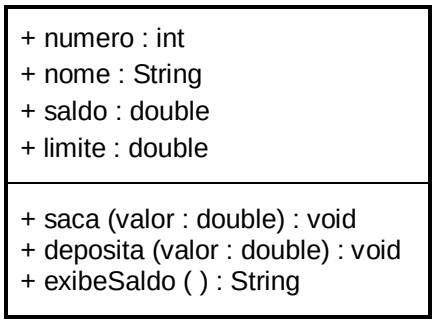
- Comandos em Java sempre com letras minúsculas.

4.2. CRIAÇÃO DE CLASSES EM JAVA

O conceito de classe apresentado anteriormente é genérico e pode ser aplicado em diversas linguagens de programação. Mostraremos como as classes poderiam ser escritas utilizando a linguagem Java.

Para apresentar os conceitos, simularemos a criação de um sistema bancário. Percebemos com facilidade que uma entidade extremamente importante para o nosso sistema é a conta. Nossa idéia aqui é generalizarmos alguma informação, juntamente com funcionalidades que toda conta deve ter. Como o nosso objetivo é criar um sistema orientado a objetos, devemos utilizar classes para representar os elementos do sistema bancário.

Conta

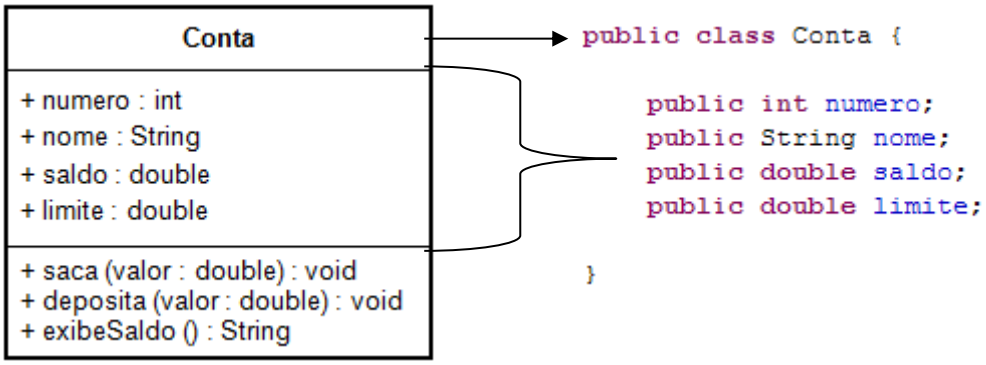


Uma classe funciona como uma forma para criar objetos. Inclusive, vários objetos podem ser criados a partir de uma única classe. Assim como vários prédios poderiam ser construídos a partir de uma única planta.

Os objetos criados a partir da classe **Conta** terão todos os atributos e métodos mostrados no diagrama UML. As diferenças entre esses objetos são os valores dos atributos.

4.5.1. Criação da classe Conta

Vamos começar apenas com o que uma Conta tem, e não com o que ela faz (veremos logo em seguida). Um tipo desses, como a especificação de Conta acima, pode ser facilmente traduzido para Java:



A classe Java **Conta** é declarada utilizando a palavra reservada **class**. No corpo dessa classe, são declaradas quatro variáveis que são os atributos que os objetos possuirão. Como a linguagem Java é estaticamente tipada os tipos dos atributos são definidos no código. O atributo saldo e limite são do tipo double que permite armazenar números com casas decimais e o atributo numero é do tipo int que permite armazenar números inteiros.

4.5.2. Criação de Objetos em Java

Após definir a classe `conta`, podemos criar objetos a partir dela. Esses objetos devem ser alocados na memória RAM do computador. Felizmente, todo o processo de alocação do objeto na memória é gerenciado pela máquina virtual. O gerenciamento da memória é um dos recursos mais importantes oferecidos pela máquina virtual.

Do ponto de vista da aplicação, basta utilizar um comando especial para criar objetos que a máquina virtual se encarrega do resto. O comando para

criar objetos é o `new`, utilizamos também os parênteses, que descobriremos o que são, exatamente, em um capítulo posterior:

```
1
2  public class Principal {
3
4      public static void main(String[] args) {
5
6          new Conta();
7
8      }
9
10 }
```

A linha com o comando new poderia ser repetida cada vez que desejássemos criar (instanciar) um objeto da classe **Conta**. A classe Java **Principal** serve apenas para colocarmos o método main que é o ponto de partida da aplicação.

Chamar o comando **new** passando uma classe Java é como se estivéssemos contratando uma construtora e passando a planta da casa que queremos construir. A construtora se encarrega de construir a casa para nós de acordo com a planta.

4.5.3. Referência de Objetos

Após criar um objeto, provavelmente, desejaremos utilizá-lo. Para isso precisamos acessá-lo de alguma maneira. Os objetos são acessados através de referências. Uma referência é "link" que aponta para um objeto.

Ao utilizar o comando **new**, um objeto é alocado em algum lugar da memória RAM. Para que possamos acessar esse objeto, precisamos da referência dele. O comando **new** devolve a referência do objeto que foi criado.

Para guardar as referências devolvidas pelo comando **new**, podemos declarar variáveis cujo propósito é justamente armazenar referências.

```
1
2  public class Principal {
3
4      public static void main(String[] args) {
5
6          Conta minhaConta;
7
8          minhaConta = new Conta();
9
10     }
11
12 }
```

Pode parecer estranho escrevermos duas vezes **Conta**: uma vez na declaração da variável e outra vez no uso do **new**. Mas há um motivo, que entenderemos também posteriormente. Através da variável **minhaConta**, agora, podemos acessar o objeto recém criado para alterar seu nome, seu saldo, etc.

4.5.4. Manipulando Atributos

Podemos alterar os valores dos atributos de um objeto se tivermos a referência dele. Os atributos são acessados pelo nome. No caso específico da linguagem Java, a sintaxe para acessar um atributo utiliza o operador ".".

Principal.java

```
1  import javax.swing.JOptionPane;
2
3  public class Principal {
4
5      public static void main(String[] args) {
6
7          Conta minhaConta;
8
9          minhaConta = new Conta();
10
11         minhaConta.nome = "Horácio";
12         minhaConta.saldo = 1000.0;
13
14         JOptionPane.showMessageDialog(null, "Saldo atual: " + minhaConta.saldo);
15
16     }
17
18 }
19
```

No código acima, o atributo NOME recebe o valor Horácio. O atributo SALDO recebe o valor 1000. Depois, os valores são impressos na tela através do comando showMessageDialog.

4.3. EXERCÍCIOS: Criação de classes de modelagem

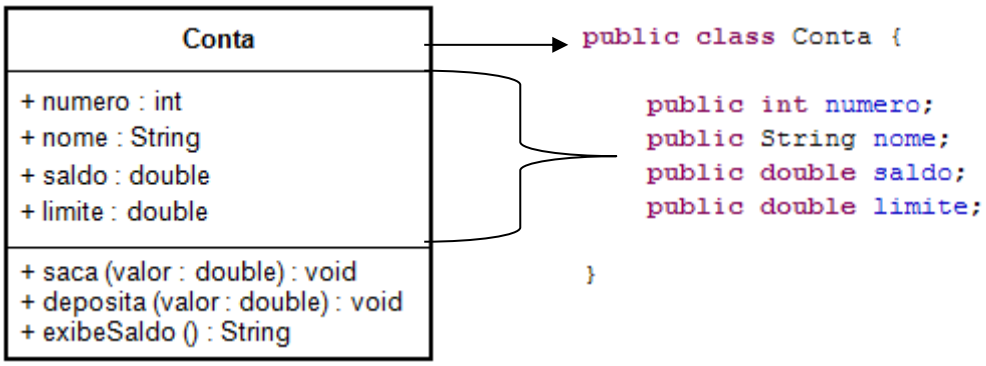
1) Vá em **`File`** ➔ **`New`** ➔ **`Project`**.

2) Selecione `Java Project` e clique em **Next**;

3) Coloque o nome do projeto como *`ds1-controlebancario`* e clique em **Finish**;

4) Não aceite a mudança de perspectiva.

5) Crie a classe a seguir:



6) Crie a classe Principal com o método *public static void main()*.

7) Instancie um objeto chamado minhaConta.

8) Altere os atributos numero, nome, saldo e limite com os valores digitados pelo usuário. Utilize a classe JOptionPane, método showInputDialog para atribuir os dados.

9) Exiba os atributos, conforme figura abaixo: