1. Aula 7

Orientação a objetos

Programas grandes são difíceis de manter, por isso é um bom hábito separá-los em unidades mais ou menos isoladas. Em Java, isso é feito utilizando objetos, que são compostos por atributos e métodos definidos a partir de classes que, por sua vez, são organizadas em pacotes. Esses conceitos são tão centrais em Java que não se pode programar na linguagem sem utilizá-los.

Todo programa em Java usa classes e objetos, por isso compreender esses conceitos é fundamental para entender a própria linguagem. Na prática, sistemas de software reais são grandes e precisam ser fatorados em partes relativamente independentes para serem viáveis. Como em Java isso é feito com classes e objetos, compreendê-los é imprescindível para escrever qualquer programa significativo.

1.1. Classes

Em Java, os programas são escritos em pequenos pedaços separados, chamados de objetos. Objetos são pequenos programas que guardam dentro de si os dados, em suma, as variáveis que precisam para executar suas tarefas. Os objetos também trazem em si, como sub-rotinas, as instruções para processar esses dados. As variáveis que um objeto guarda são chamadas de atributos, as suas sub-rotinas, de métodos.

Em Java, objetos são criados a partir de modelos que os descrevem. Esses modelos são chamados de classes. É dentro dessas classes que definimos que atributos os objetos conterão e que métodos os objetos fornecerão.

Exemplo:

Vamos começar apenas com o que uma **Conta** tem, e não com o que ela faz (veremos logo em seguida).

Um tipo desses, como o especificado de Conta acima, pode ser facilmente traduzido para Java:

```
class Conta {
   int numero;
   String titular;
   double saldo;
}
```

String

String é uma classe em Java. Ela guarda uma cadeia de caracteres, uma frase completa.

Por enquanto, declaramos o que toda conta deve ter. Estes são os atributos que toda conta, quando criada, vai ter.

Características das classes

-Toda classe possui um nome;

-Possuem visibilidade, exemplo: public, private, protected;

-Possuem membros como: Características e Ações;

-Para criar uma classe, basta declarar a visibilidade + digitar a palavra reservada class + NomeDaClasse + abrir e fechar chaves { }.

```
public class Teste{
        //ATRIBUTOS OU PROPRIEDADES
        //MÉTODOS
}
```

1.2. Métodos

Dentro da classe, também declararemos o que cada conta faz e como isso é feito, os comportamentos que cada classe tem, isto é, o que ela faz. Por exemplo, de que maneira que uma Conta saca dinheiro? Especificaremos isso dentro da própria classe Conta, e não em um local desatrelado das informações da própria Conta. É por isso que essas "funções" são chamadas de métodos, pois é a maneira de fazer uma operação com um objeto.

Queremos criar um método que saca uma determinada quantidade e não devolve nenhuma informação para quem acioná-lo:

```
class Conta {
   double salario;
   // ... outros atributos ...

   void saca(double quantidade) {
      double novoSaldo = this.saldo - quantidade;
      this.saldo = novoSaldo;
   }
}
```

A palavra-chave void diz que, quando você pedir para a conta sacar uma quantia, nenhuma informação será enviada de volta a quem pediu.

Quando alguém pedir para sacar, ele também vai dizer quanto quer sacar, por isso precisamos declarar o método com algo dentro dos parênteses - o que vai aí dentro é chamado de argumento do método (ou parâmetro). Essa variável é uma variável comum, chamada também de temporária ou local, pois, ao final da execução desse método, ela deixa de existir.

Dentro do método, estamos declarando uma nova variável. Ela, assim como o argumento, vai morrer no fim do método, pois este é seu escopo. No momento que vamos acessar nosso atributo, usamos a palavra-chave this para mostrar que esse é um atributo, e não uma simples variável. (veremos depois que é opcional)

Métodos com retorno

Um método sempre tem que definir o que retorna, nem que defina que não há retorno, como nos exemplos anteriores onde estávamos usando o void.

Um método pode retornar um valor para o código que o chamou. No caso do nosso método saca, podemos devolver um valor booleano indicando se a operação obteve sucesso.

```
class Conta {
   // ... outros métodos e atributos ...

   boolean saca(double valor) {
      if (this.saldo < valor) {
         return false;
      }
      else {
         this.saldo = this.saldo - valor;
         return true;
      }
   }
}
```

A declaração do método mudou! O método saca não tem void na frente. Isso quer dizer que, quando é acessado, ele devolve algum tipo de informação. No caso, um boolean. A palavra-chave return indica que o método vai terminar ali, retornando tal informação.

Exemplo de uso:

```
minhaConta.saldo = 1000;
boolean consegui = minhaConta.saca(2000);
if (consegui) {
   System.out.println("Consegui sacar");
} else {
   System.out.println("Não consegui sacar");
}
```

1.3. Herança

O que é herança na vida real?

É quando uma pessoa deixa seus bens para outra. No Java também.

Geralmente a herança ocorre entre membros de uma mesma família. No Java também.

Herança, em Java, nada mais é do que criar classes usando outras já existentes.

Obviamente, você vai fazer uma classe herdar as características de outra se essas tiverem uma relação (se forem parecidas).

Outro ponto importante é que, quando fazemos uso da herança, nós podemos adicionar mais atributos à classe.

Exemplo 1: Carros e motos

Imagine que você tem uma revenda de veículos: carros e motos.

Todos são veículos. Ora, crie a classe "Veículo".

O que esses veículos têm em comum?

Motor, preço, marca, nome do cliente que vai comprar, quantos quilômetros fazem com 1 litro de combustível, etc.

Todos os objetos da classe “Veículo” têm essas características.

No entanto, existem algumas características que as motos têm que um carro não tem: capacete, somente duas rodas, cilindrada, etc.

Também existem características que os carros têm que as motos não têm: podem ter 4 portas, banco de couro, ar-condicionado, etc.

Para resolver esse problema e deixar a aplicação MUITO MAIS ORGANIZADA, vamos fazer duas classes: "Moto" e "Carro".

Cada uma dessas irá herdar a classe "Veículo", pois também são veículos.

Dizemos que "Veículo" é a superclasse a, "Moto" e "Carro" são subclasses.