Java Basics

Define the scope of variables

O escopo é o que determina em que pontos do código uma variável pode ser usada.

Variáveis locais

Chamamos de locais as variáveis declaradas dentro de métodos ou construtores. Antes de continuar, vamos estabelecer uma regra básica: o ciclo de vida de uma variável local vai do ponto onde ela foi declarada até o fim do bloco onde ela foi declarada.

Mas o que é um bloco? Podemos entender como bloco um trecho de código entre chaves. Pode ser um método, um construtor, o corpo de um ***if***, de um ***for*** etc.:

```
public void m1() { // inicio do bloco do metodo
    int x = 10; // variavel local do metodo

    if (x >= 10) { // inicio do bloco do if
        int y = 50; // variavel local do if
        System.out.print(y);
    } // fim do bloco do if
} // fim do bloco do metodo
```

Analisando esse código, temos uma variável ***x***, que é declarada no começo do método. Ela pode ser utilizada durante todo o corpo do método. Dentro do ***if***, declaramos a variável ***y***. ***y*** só pode ser utilizada dentro do corpo do ***if***, delimitado pelas chaves. Se tentarmos usar ***y*** fora do corpo do ***if***, teremos um erro de compilação, pois a variável saiu do seu escopo.

Tome cuidado especial com loops ***for***. As variáveis declaradas na área de inicialização do loop só podem ser usadas no corpo do loop:

```
    for (int i = 0, j = 0; i < 10; i++)
    j++;

    System.out.println(j); // erro, ja nao esta mais no escopo
```

Parâmetros de métodos também podem ser considerados variáveis locais ao método, ou seja, só podem ser usados dentro do método onde foram declarados:

```
class Teste {

    public void m1(String bla) {
        System.out.print(bla);
    }

    public void m2() {
        // erro de compilacao pois bla nao existe neste
        // escopo
        System.out.println(bla);
    }
}
```

Variáveis de instância

Variáveis de instância ou variáveis de objeto são os atributos dos objetos. São declaradas dentro da classe, mas fora de qualquer método ou construtor. Podem ser acessadas por qualquer membro da classe e ficam em escopo enquanto o objeto existir:

```
class Pessoa {
    // variavel de instancia ou variavel de objeto
    String nome;

    public void setNome(String n) {
        // acessando a variavel de instancia no metodo
        this.nome = n;
    }
}
```

Variáveis estáticas (class variables)

Podemos declarar variáveis que são compartilhadas por todas as instâncias de uma classe usando a palavra chave ***static***. Essas variáveis estão no escopo da classe, e lá ficarão enquanto a classe estiver carregada na memória (enquanto o programa estiver rodando, na grande maioria dos casos).

```
class Pessoa {
    static int id = 1;
}

class Teste {
    public static void main(String[] args) {
        Pessoa p = new Pessoa();
        System.out.println(p.id); // acessando pelo objeto
        System.out.println(Pessoa.id); // acessando direto pela
        // classe
    }
}
```

No caso de variáveis ***static***, não precisamos ter uma referência para usá-las e podemos acessá-las diretamente a partir da classe, desde que respeitando as regras de visibilidade da variável.

Variáveis com o mesmo nome

Logicamente, não é possível declarar duas variáveis no mesmo escopo com o mesmo nome:

```
public void bla() {
    int a = 0;
    int a = 10;// erro
}
```

Mas, eventualmente, podemos ter variáveis em escopos diferentes que podem ser declaradas com o mesmo nome. Em casos em que possa haver ambiguidade na hora de declará-las, o próprio compilador irá emitir um erro evitando a confusão.

Por exemplo, não podemos declarar variáveis de classe e de instância com o mesmo nome:

```
    class Bla {
        static int a;
        int a; // erro de compilacao,
    }
    ...
    System.out.println(new Bla().a); // qual variavel estamos
    // acessando?
```

Também não podemos declarar variáveis locais com o mesmo nome de parâmetros:

```
    public void metodo(String par) {
        int par = 0; // erro de compilacao

        System.out.println(par); // qual?
    }
```

Apesar de parecer estranho, é permitido declarar variáveis locais ou parâmetros com o mesmo nome de variáveis de instância ou de classe. Essa técnica é chamada de *shadowing*. Nesses casos, é possível resolver a ambiguidade: para variáveis de classe, podemos referenciar pela própria classe; para variáveis de instância, usamos a palavra chave ***this***:

```
class Pessoa {

    static int x = 0;
    int y = 0;

    public static void setX(int x) {
        // Usando a referencia da classe
        Pessoa.x = x;
    }

    public void setY(int y) {
        // usando o this
        this.y = y;
    }
}
```

Quando não usamos o ***this*** ou o nome da classe para usar a variável, o compilador sempre utilizará a variável de menor escopo:

```
class X {
    int a = 10;

    public void metodo() {
        int a = 20; // shadowing
        System.out.println(a); // imprime 20
    }
}
```

Define the structure of a Java class

Nesta seção, iremos entender a estrutura de um arquivo java, onde inserir as declarações de pacotes e imports e como declarar classes e interfaces.

Para entender a estrutura de uma classe, vamos ver o arquivo ***Pessoa.java***:

```
    // Declaracao de pacote
    package br.com.caelum.certificacao;

    // imports
    import java.util.Date;

    // Declaracao da classe
    class Pessoa {
        // conteudo da classe
    }
```

Pacotes

Pacotes servem para separar e organizar as diversas classes que temos em nossos sistemas. Todas as classes pertencem a um pacote, sendo que, caso o pacote não seja explicitamente declarado, a classe fará parte do que chamamos de pacote padrão, ou ***default package***. Todas as classes no ***default package*** se enxergam e podem ser utilizadas entre si. Classes no pacote ***default*** não podem ser importadas para uso em outros pacotes:

```
// Uma classe no pacote padrao
class Pessoa {
    //...
}
```

Para definir qual o pacote a que a classe pertence, usamos a palavra-chave ***package***, seguida do nome do pacote. Só pode existir um único ***package*** definido por arquivo, e ele deve ser a primeira instrução do arquivo. Após a definição do ***package***, devemos finalizar a instrução com um ***;***. Podem existir comentários antes da definição de um pacote:

```
    // declaracao do pacote
    package br.com.caelum.certificacao;

    class Pessoa {
        //...
    }
```

Aproveitando que tocamos no assunto, o ***package*** deve ser a primeira instrução de código que temos declarada em nosso arquivo. Comentários não são considerados parte do código, portanto, podem existir em qualquer lugar do arquivo java sem restrições.

Para inserir comentário em nosso código, temos as seguintes formas:

```
1   // comentario de linha
2
3   /*
4   comentario de
5   multiplas linhas
6   */
7   class /* comentario no meio da linha */ Pessoa {
8
9       /**
10      *  JavaDoc, repare que a primeira linha do comentario tem
11      *  2 asteriscos
12      */
13      public void metodo() {
14      }
15  }
```

Create executable Java applications with a main method; run a Java program from the command line; produce console output

Import other Java packages to make them accessible in your code

Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.