Tipo Estático e Tipo Dinâmico

Em Java todas as variáveis têm de ser declaradas antes de serem usadas.

Podemos definir dois tipos de dados para cada variável: tipo estático e tipo dinâmico.

- Tipo estático de uma variável é o tipo da sua declaração no código fonte.
- Tipo dinâmico de uma variável é o tipo do objeto que num dado momento está atribuído à variável; depende das atribuições em tempo de execução - do comportamento dinâmico do programa.

A uma variável declarada de um dado tipo (classe) pode-lhe ser atribuído um objeto desse tipo ou de qualquer subtipo (subclasse). Esta regra, designada por Princípio da Substituição, é uma característica da Programação Orientada por Objetos.

Exemplo:

- Consideremos que as classes Pixel e Ponto3D são subclasses de Ponto.
- O tipo estático de "p1" é Ponto e o tipo dinâmico é Pixel.
- O tipo estático de "p2" é Ponto e o tipo dinâmico pode ser Ponto, Pixel ou Ponto3D. Só em tempo de execução é possível determinar o tipo dinâmico.

```
public class Ponto {}
public class Pixel extends Ponto {}
public class Ponto3D extends Ponto {}
public class Teste1 {
   public static void main(String[] args) {
      Ponto p1 = new Pixel();
      Ponto p2 = new Ponto();

      // Geração de número aleatorio [0, 1, 2]
      int i = (int) (3 * Math.random());

      if (i==1) p2 = new Pixel();
      else if (i==2) p2 = new Ponto3D();

      if (p2 instanceof Pixel)
         System.out.println("p2 é do tipo Pixel");

      if (p2 instanceof Ponto3D)
         System.out.println("p2 é do tipo Ponto3D");

      if (p2 instanceof Ponto)
         System.out.println("p2 é do tipo Ponto");
   }
}
```

- Saídas possíveis que podem ocorrer em diferentes execuções:

```
----------------------------
 p2 é do tipo Pixel
 p2 é do tipo Ponto
----------------------------
 p2 é do tipo Ponto3D
 p2 é do tipo Ponto
----------------------------

p2 é do tipo Ponto
----------------------------
```

Métodos de Instância

Um método de instância é sempre invocado para uma variável de instância.

O compilador efetua uma verificação de tipos (*type checking*) usando o tipo estático da variável. O tipo dinâmico muitas vezes só é conhecido em tempo de execução pelo que o compilador apenas pode fazer verificações relativamente ao tipo estático. O compilador verifica se o tipo estático (classe) da variável para a qual o método de instância é invocado contém esse método, ou se existe numa superclasse, e se é acessível (atendendo ao modificador de acesso do método).

No entanto, em tempo de execução, o interpretador efetua uma ligação dinâmica de métodos (*dynamic binding*): determina o tipo dinâmico da variável e procura o código do método nessa classe. Se encontrar executa esse código, se não encontrar, procura na sua superclasse e depois subindo na hierarquia de classes.

Se um programa compila com sucesso, há a garantia de que um método de instância exista no tipo estático podendo eventualmente estar reescrito em subclasses.

Uma vez que o tipo dinâmico de uma variável é um subtipo do tipo estático ou o próprio tipo estático, há a garantia de em tempo de execução, o interpretador encontrar o método, ainda que efetuando a procura a partir do tipo dinâmico. Embora só em tempo de execução o interpretador possa determinar o tipo dinâmico de uma variável e portanto o código do método a executar, este mecanismo de ligação de métodos permite escolher o método de instância mais adequado ao objeto sobre o qual o método é invocado.

Em resumo:

Na compilação, a verificação de tipos (*type checking*) usa o tipo estático (*static type*), mas na execução os métodos do tipo dinâmico (*dynamic type* ou *run time type*) são executados.

A procura de um método em tempo de execução, por exemplo em *v.m(),* é efetuada nos seguintes passos:

1. A variável *v* é acedida,
2. O objeto armazenado nessa variável é encontrado.
3. A classe do objeto é encontrada.
4. A implementação do método *m()*, se encontrada na classe, é executada.
5. Se o método *m()* não existe na classe, é procurado na superclasse, subindo na hierarquia de herança até ser encontrado. Durante a execução, é seguro que o método será encontrado, porque senão a classe não teria compilado.

Variáveis de Instância, Variáveis Estáticas e Métodos Estáticos

Para variáveis de instância, variáveis estáticas e métodos estáticos o tipo estático é que determina o valor ou método a usar, como se pode ver no exemplo seguinte:

```
public class A {
    int a = 3;
    static int sa = 5;
    public static String m() { return "Método da classe A"; }
}

public class B extends A {
    int a = 33;
    static int sa = 55;
    public static String m() { return "Método da classe B"; }
}

public class Teste2 {
    public static void main(String[] args) {

      A a1 = new B();

      System.out.println("Var. ref. do tipo A contém obj. de tipo B: a1");

      System.out.println("Variável de instância: a1.a = " + a1.a );

      System.out.println("Variável estática: a1.sa  = " + a1.sa );

      System.out.println("Método estático: a1.m() = " + a1.m() );
    }
}
```

Saída produzida:

```
Var. ref. do tipo A contém obj. de tipo B: a1

Variável de instância: a1.a = 3

Variável estática: a1.sa = 5

Método estático: a1.m() = Método da classe A
```