Classes internas

Java
Classes internas

Classes internas

- Classes podem ser membros de classes, de objetos ou locais a métodos. Podem até serem criadas sem nome, apenas com corpo no momento em que instanciam um objeto
  - Há poucas situações onde classes internas podem ou devem ser usadas.
  - Usos típicos incluem tratamento de eventos em GUIs, criação de threads, manipulação de coleções e sockets
- Classes internas podem ser classificadas em 4 tipos
  - Classes estáticas - classes membros de classe
  - Classes de instância – classes membros de objetos
  - Classes locais – classes dentro de métodos
  - Classes anónimas - classes dentro de instruções

Classes estáticas

- São declaradas como *static* dentro de uma classe
- A classe externa age como um pacote para uma ou mais classes internas estáticas
  - Externa.Coisa, Externa.InternaUm, ..
- O compilador gera arquivos tipo *Externa$InternaUm.class*

```
class Externa {
   private static class InternaUm {
      public int campo;
      public void metodoInterno() {...}
   }
   public static class InternaDois
                        extends InternaUm {
      public int campo2;
      public void metodoInterno() {...}
   }
   public static interface Coisa {
      void existe();
   }
   public void metodoExterno() {...}
}
```

Classes de instância

- São membros do objeto, como métodos e atributos
- Requerem que objeto exista antes que possam ser usadas.
  - Externamente usa-se referência.new para criar objetos
- Deve usar-se *NomeDaClasse.this* para aceder a campos internos

```
class Externa {
   public int campoUm;
   public class Interna {
     public int campoUm;
     public int campoDois;
     public void metodoInterno() {
         this.campoUm = 10;          // Externa.campoUm
         Interna.this.campoUm = 15;
     }
   }
   public static void main(String[] args) {
     Interna e = (new Externa()).new Interna();
} }
```

Classes locais

- Servem para tarefas temporárias já que deixam de existir quando o método acaba
  - Têm o mesmo alcance de variáveis locais.

```
public Multiplicavel calcular(final int a, final int b) {
    class Interna implements Multiplicavel {
        public int produto() {
            return a * b; // usa a e b, que são constantes
        }
    }
    return new Interna();
}
public static void main(String[] args){
    Multiplicavel mul = (new Externa()).calcular(3,4);
    int prod = mul.produto();
}
```

Classes anónimas

- Servem para criar um único objeto
  - A classe abaixo estende ou implementa SuperClasse, que pode ser uma interface ou classe abstracta.

```
Object i = new SuperClasse() {
    // implementação
};
```

```
public Multiplicavel calcular(final int a, final int b) {
    return new Multiplicavel()  {
        public int produto() {
            return a * b;
        }
    };
}
public static void main(String[] args){
    Multiplicavel mul = (new Externa()).calcular(3,4);
    int prod = mul.produto();
}
```

Compare com parte em preto e vermelho do slide anterior!

A classe está dentro da instrução: preste atenção no ponto-e-vírgula!