Paradigmas de Programação

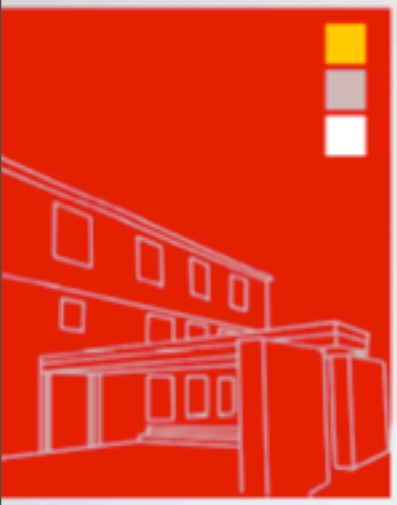
Aula 07

1. Palavras Reservadas
2. Herança
3. Overriding
4. Palavras Reservadas Usadas
5. Links Úteis

Palavras reservadas usadas

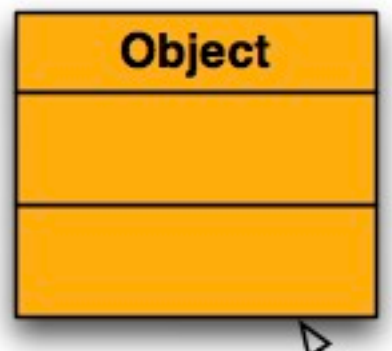
abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

* not used
** added in 1.2
*** added in 1.4
**** added in 5.0



Conceitos

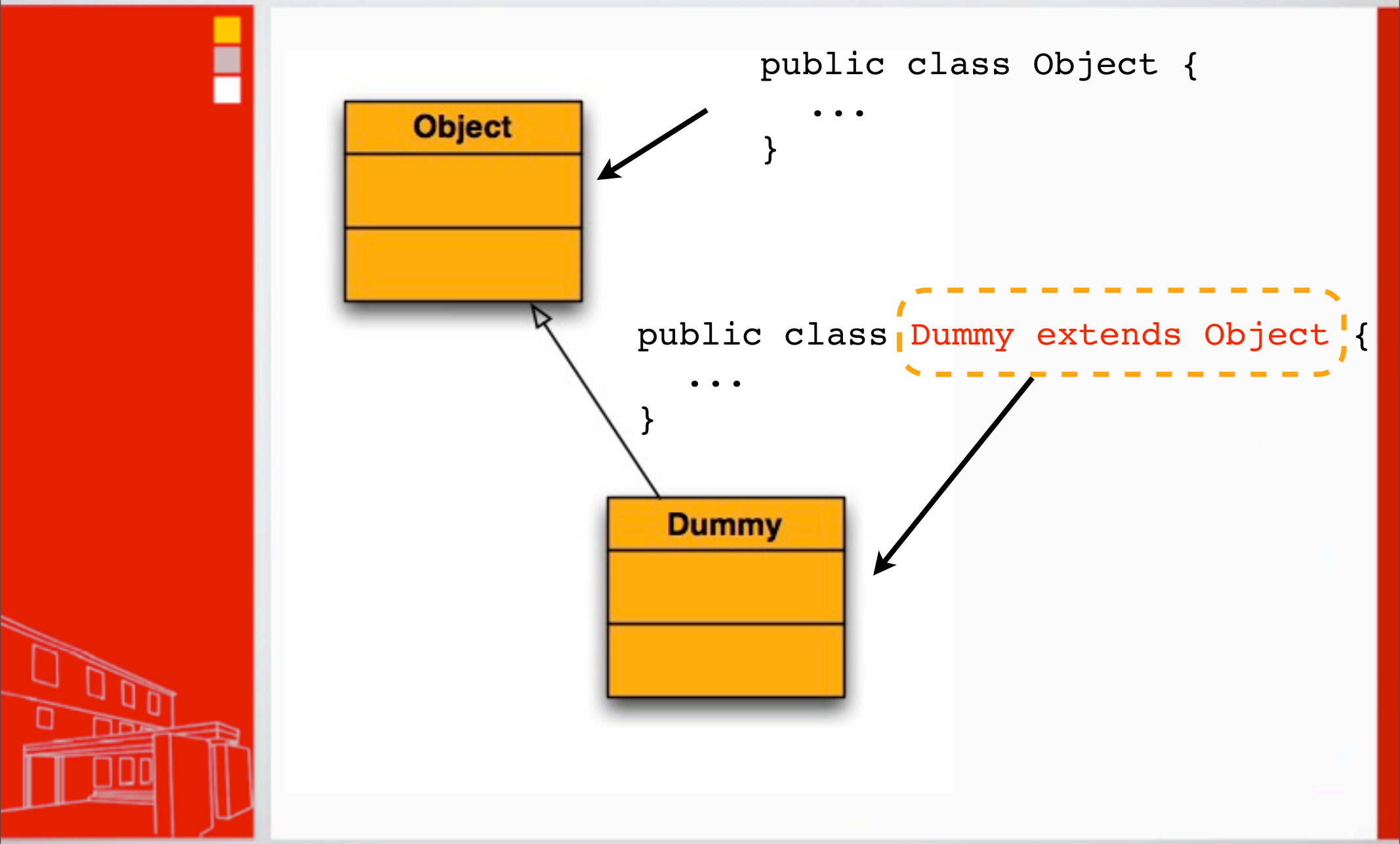
- **Superclass**
  - *Class* que é uma generalização das suas classes descendentes directas ou não
- **Subclass**
  - *Class* que é uma especialização daquele que descende
  - Em *Java* não há herança múltipla, ou seja, uma *class* descende de uma única (`Object` não possui superclass)

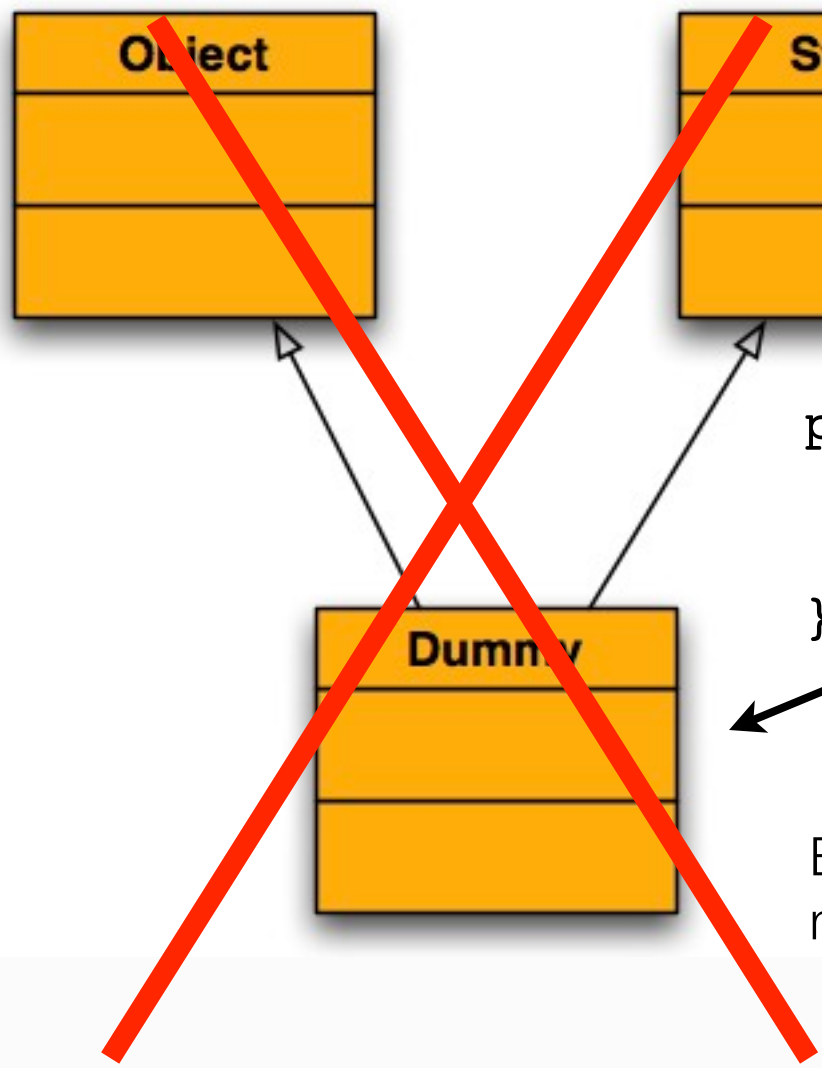


Superclass
Class genérica

Em Java esta associação é explicitada através da keyword (palavra reservada) **extends**

Subclass
Class especializada herda todos os membros (atributos, métodos e classes "aninhadas") da superclasse


public class Object {
...
}
Object
public class Dummy extends Object {
...
}
Dummy

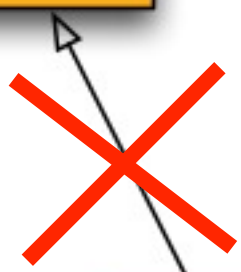



```
public class Dummy
  extends Object, String {
    ...
}
```

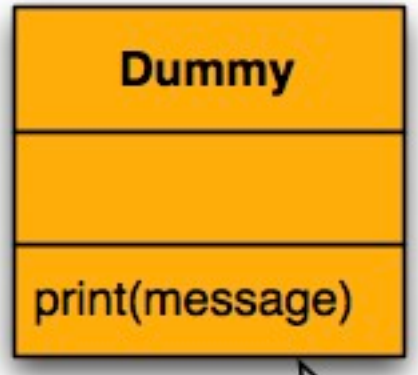
Em Java **não** é possível herança múltipla!


```
public final class Dummy {
   ...
}
```

```
public class MyDummy
     extends Dummy {
        ...
}
```



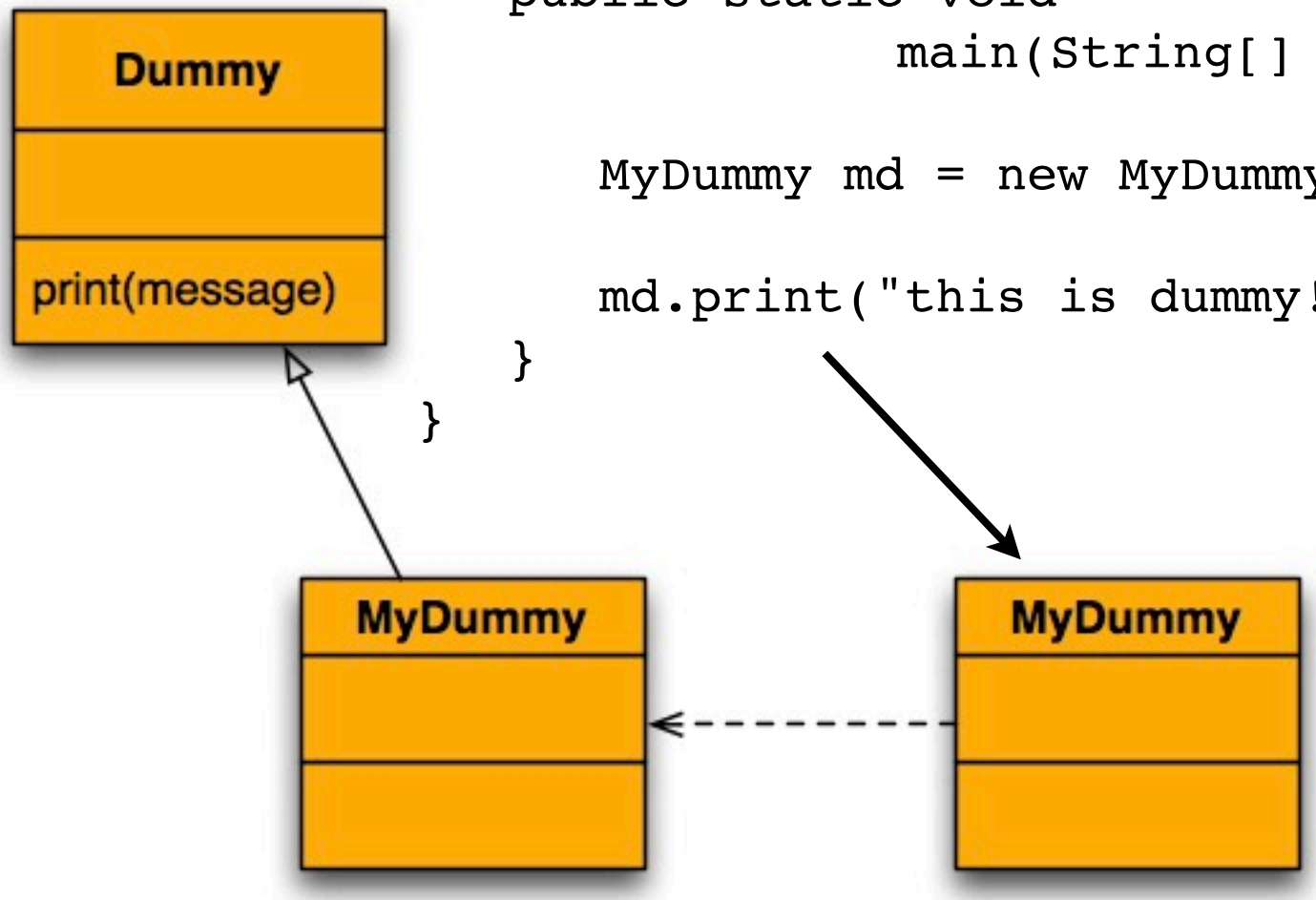
`MyDummy` **não** pode descender de `Dummy`, uma vez que `Dummy` foi declarada como **final**




```
public class Dummy {
   void print(String message) {
       System.out.println(message);
   }
}
```

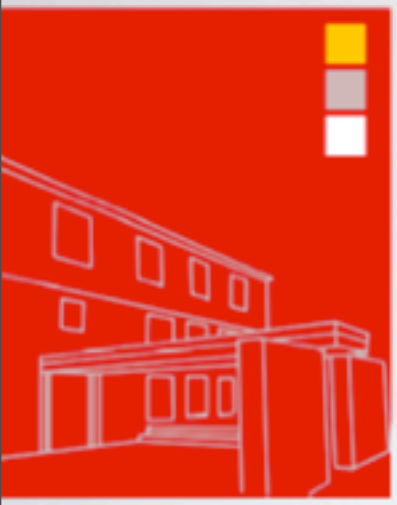
`MyDummy` herda o método **print**, o qual foi assinado na superclass

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

        MyDummy md = new MyDummy();

        md.print("this is dummy!");
    }
}
```



- A execução da **`class Test`** produz o seguinte resultado:

```
this is dummy!
```

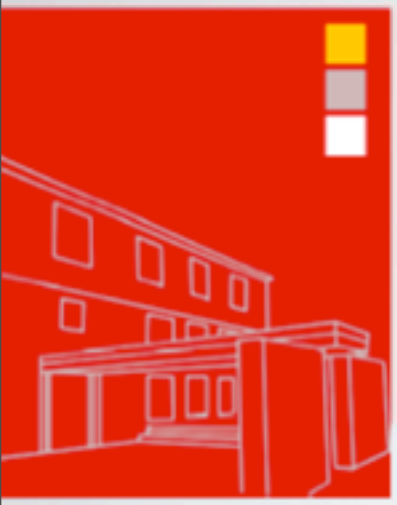


Overriding

- O que fazer se desejarmos que o comportamento de **print** seja outro?

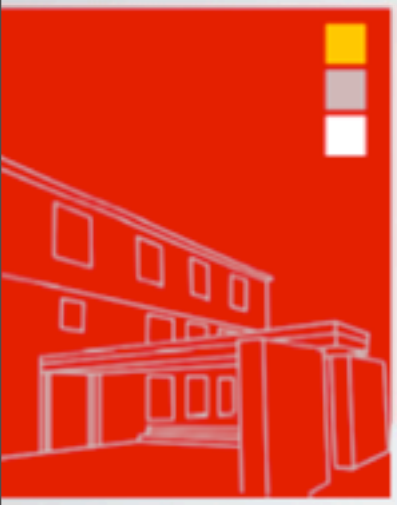
- Por exemplo:

```
**********
this is dummy!
**********
```



Solução do tipo “força-bruta”

```
public class Test {
   public static void main (String[] args) {
      MyDummy md = new MyDummy();

      md.print("***************");
      md.print("this is dummy!");
      md.print("***************");
   }
}
```

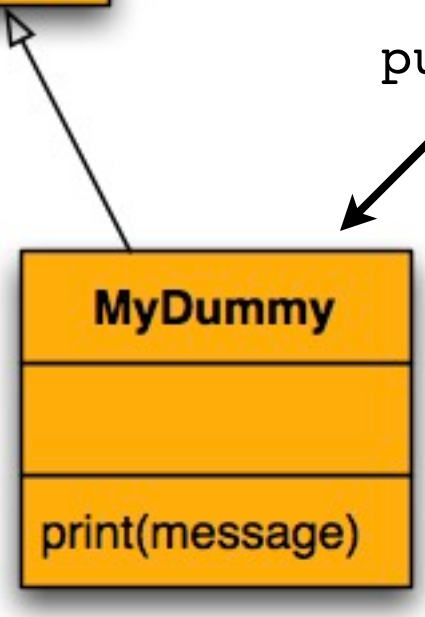


Solução elegante

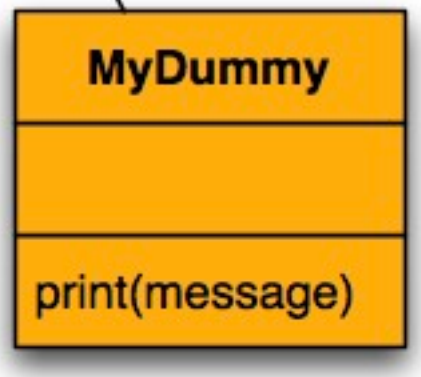
- A solução elegante recorre a uma funcionalidade de programação orientada aos objectos:

- **Overriding**


```
public class Dummy {
   void print(String message) {
      System.out.println(message);
   }
}
```




```
public class MyDummy
      extends Dummy {

   @Override
   void print(String message) {
      System.out.println(
         "**********\n" +
         message+
         "\n**********");
   }
}
```



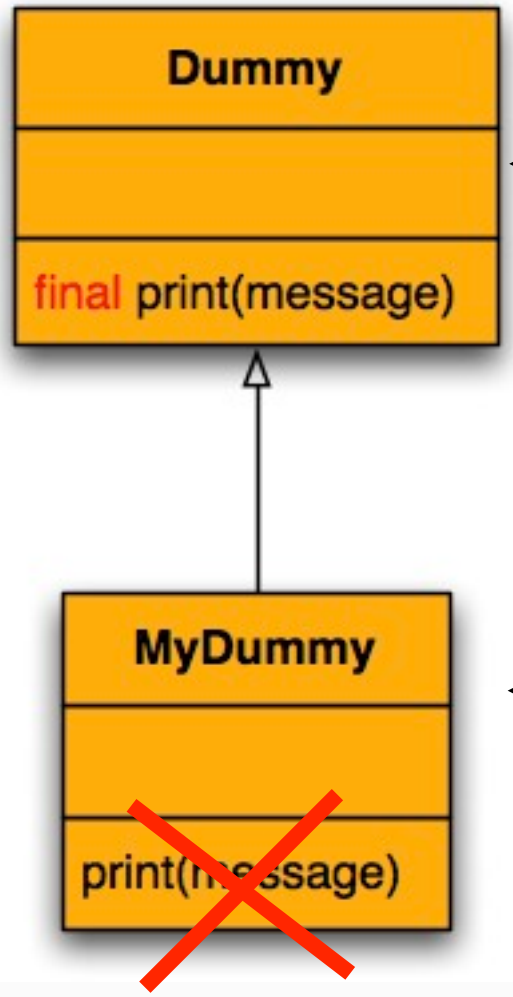

```
public class Dummy {
   void print(String message) {
      System.out.println(message);
   }
}
      public class MyDummy
            extends Dummy {

   @Override
   void print(String message) {
      System.out.println(
         "**********");
      super.print(message);
      System.out.println(
         "**********");
      }
}
```

Conceito:

- Um método que faça **overriding**, que se sobreponha, a um herdado não pode diminuir a visibilidade em relação ao herdado

- Se o herdado for **protected** o que sobrepõe **não** pode ser **private**

- Conceito:
  - Não é possível fazer **overriding** de um método que foi assinado como **final**



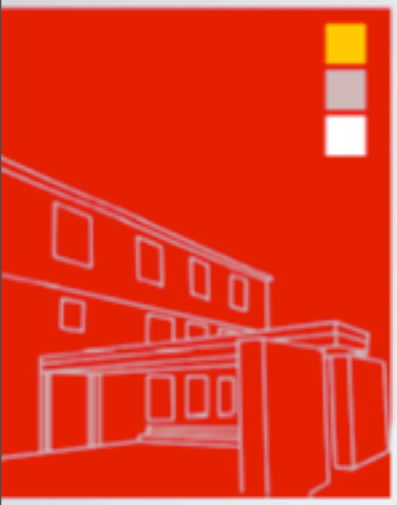

```
public class Dummy {
   void print(String message) {
      System.out.println(message);
   }
}
```

```
public class MyDummy
      extends Dummy {

   @Override
   void print(String message) {
      ...
   }
}
```

Palavras reservadas usadas

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

* not used
** added in 1.2
*** added in 1.4
**** added in 5.0



Links Úteis

- http://java.sun.com/docs/books/tutorial/java/landl/subclasses.html
- http://java.sun.com/docs/books/tutorial/java/landl/override.html