Linguagens de Programação II

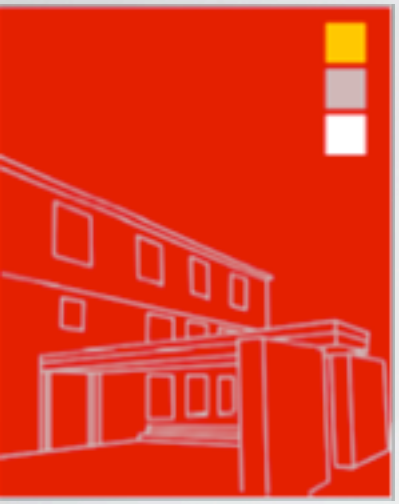
Aula 04

1. Palavras Reservadas
2. Criação de Objectos
3. Métodos Construtores
4. Variáveis de Instância
5. Strings
6. Palavras Reservadas Usadas
7. 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



Objecto

- Um objecto é uma instância de uma classe.
- Construído a partir da especificação de uma classe.
- Com uma identidade única.
- Em java para criarmos um objecto usamos o operador de alocação **`new`** como podemos ver no slide seguinte.

- Os objectos são alocados dinamicamente e o operador de alocação é o **`new`**:

Sintaxe de alocação
`new <data-type>(<arguments>...)`
`<data-type> <variable> = new <data-type>(<arguments>...)`

```
public class Dog {

  char[] name = {'f', 'i', 'd', 'o'};
  char[] bark = {'w', 'o', 'o', 'f','!'};
  int age = 6;
}
```

Os atributos `name`, `bark` e `age` da classe `Dog` são denominados variáveis de instância.

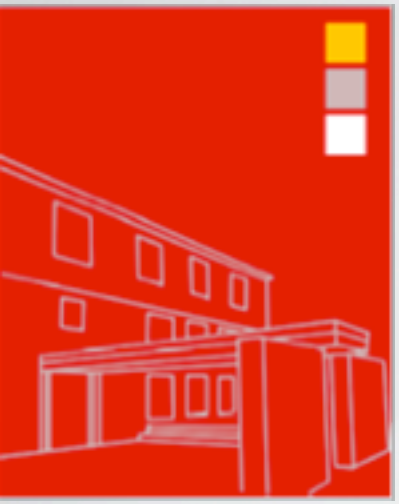
```
class ADogsLife {

   public static void main(String[] args) {

      Dog fido = new Dog();

      System.out.println(fido.name);
      System.out.println(fido.bark);
      System.out.println(fido.age);

   }

}
```

new é o operador de alocação.

Dog() é o método constructor.

Reparou que não existe o membro "Dog( )"

```
class ADogsLife {

   public static void main(String[] args) {

      Dog fido = new Dog();
      Dog spot = new Dog();

      System.out.println(fido.name);
      System.out.println(fido.bark);
      System.out.println(fido.age);

      System.out.println(spot.name);
      System.out.println(spot.bark);
      System.out.println(spot.age);

   }
}
```



Métodos Construtores

- Os construtores de uma classe são todos os métodos especiais que são declarados na classe tendo por identificador o nome exacto da classe.

- Os métodos construtores podem ter argumentos de qualquer tipo de dados e cujo objectivo é criar instâncias de tal classe que sejam de imediato manipuláveis.
- Os construtores, dado criarem instâncias de uma dada classe, não têm obviamente, que especificar qual o resultado, pois será sempre uma instância da respectiva classe.
- É possível e útil construir mais do que um construtor de instâncias de uma dada classe.

```
public class Dog {

  char[] name = {'f', 'i', 'd', 'o'};
  char[] bark = {'w', 'o', 'o', 'f', '!'};
  int age = 6;

  Dog(char[] nametmp)
  {
     name = nametmp;
  }
}
```

`Dog(char[] nametmp)` é o método construtor

```
class ADogsLife {

   public static void main(String[] args) {

      char[] fidoname = {'f', 'i', 'd', 'o'};
      Dog fido = new Dog(fidoname);
      char[] spotname = {'s', 'p', 'o', 't'};
      Dog spot = new Dog(spotname);

      System.out.println(fido.name);
      System.out.println(fido.bark);
      System.out.println(fido.age);

      System.out.println(spot.name);
      System.out.println(spot.bark);
      System.out.println(spot.age);

   }
}
```

```
public class Dog {

  char[] name = {'f', 'i', 'd', 'o'};
  char[] bark = {'w', 'o', 'o', 'f', '!'};
  int age = 6;

  Dog(char[] nametmp)
  {
     name = nametmp;
  }

  Dog(char[] nametmp, char[] barktmp, int agetmp)
  {
     name = nametmp;
     bark = barktmp;
     age = agetmp;
  }
}
```

Criámos um novo método construtor.

```
class ADogsLife {

   public static void main(String[] args) {

      char[] dogname={'s', 'p', 'o', 't'};
      char[] dogbark={'r', 'u', 'f', 'f', '!'};
      Dog spot = new Dog(dogname, dogbark,3);

      System.out.println(spot.name);
      System.out.println(spot.bark);
      System.out.println(spot.age);

   }
}
```

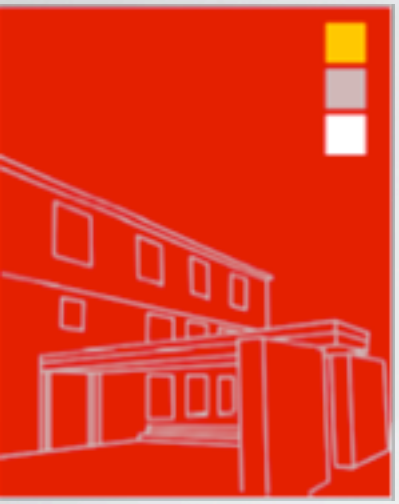
Variáveis de Instância

- Até agora aprendemos a definir instâncias de objectos mas cada objecto pode também ter variáveis de estado.
- Para cada instância de `Dog` do exemplo anterior podemos ter alguma variabilidade de objectos de acordo com as suas características tais como cor do pêlo, peso, etc.
- A estas variáveis damos o nome de variáveis de instância.

```
public class Dog {

    char[] name;
    char[] bark = {'w', 'o', 'o', 'f', '!'};
    int age = 6;

    Dog(char[] nametmp)
    {
       name = nametmp;
    }

    Dog(char[] nametmp, char[] barktmp, int agetmp)
    {
       name = nametmp;
       bark = barktmp;
       age = agetmp;
    }
}
```

Variáveis de Instância

Métodos Construtores.



String

- A `String` é um tipo não primitivo que está definido numa classe de sistema do Java - `java.lang`
- O *package* `lang` é considerado tão essencial que não é necessário indicar o caminho das classes que pretendemos utilizar deste *package*.

- Numa rápida visualização do `java.lang.String` podemos ver que existe um número muito extenso de métodos para a manipulação de uma *string*.
- As instâncias de `String` são imutáveis, isto é, não podem ser alteradas.

- Uma normal operação de display envolve *strings*:

```
System.out.println(“x= ” + x);
```

- Como o "+" é reconhecido pelo compilador de Java como um operador de concatenação de *strings*, o compilador automaticamente gera o código necessário para converter qualquer operando que não seja uma `String` numa instância de `String`.

- Como **`String`** é uma classe a forma geral para criar a instância de uma *string* é a seguinte:

```
String prompt = new String(“x= ”);
```

- De modo a ser mais cómodo para o programador a linguagem Java reconhece uma sequência de caracteres entre aspas como uma constante **`String`**
- Podemos então criar uma instância de **`String`** de uma forma mais rápida:

```
String prompt = “x= ”;
String barksound = “woof!”;
```

- Vamos alterar a nossa Classe **`Dog`** de forma a usar uma **`String barksound`** em vez do *array* de **`char bark`**.
- Alterar esta linha:

```
char[] bark = {'w','o','o','f','!'};
```

- Por esta:

```
String barksound = "woof!";
```

```
public class Dog {

    char[] name;
    String barksound = “woof!”;
    int age = 6;

    Dog(char[] nametmp)
    {
       name = nametmp;
    }

    Dog(char[] nametmp, String barktmp, int agetmp)
    {
       name = nametmp;
       barksound =barktmp;
       age = agetmp;
    }
}
```

Variáveis de Instância

Métodos Construtores.

- A definição de **`Dog`** agora inclui a variável de instância **`barksound`**.
- Cada vez que é criada uma nova instância de **`Dog`** vai incluir uma referência para a instância da **`String`** que representa o ladrar do cão.
- A linha

```
String barksound = “woof!”;
```

- aloca a instância de uma **`String`** e inicializa-a com o valor de "**`Woof`**!".

- De seguida podemos alterar também a variável de instância **`name`**:

```
char[] name;
```

- por:

```
String name;
```

```
public class Dog {

  String name;
  String barksound = “woof!”;
  int age = 6;

  Dog(String nametmp)
  {
     name=nametmp;
  }

  Dog(String nametmp, String barktmp, int agetmp)
  {
     name = nametmp;
     barksound = barktmp;
     age = agetmp;
  }
}
```

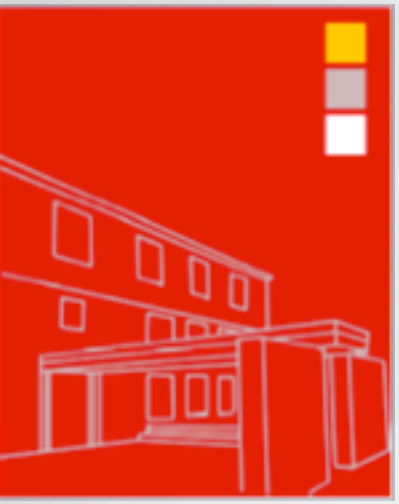
Variáveis de Instância

Métodos Construtores.

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://docs.oracle.com/javase/tutorial/java/javaOO/objects.html
- http://docs.oracle.com/javase/tutorial/java/javaOO/objectcreation.html
- http://docs.oracle.com/javase/tutorial/java/javaOO/variables.html
- http://docs.oracle.com/javase/tutorial/java/data/strings.html