Pacotes Padrões do Java

José Macedo

Técnicas de Programação

Pacote Padrão lang

- Classes centrais da linguagem
  - **Object**
  - classes para tipos básicos
  - **String** & **StringBuffer**
  - **System**
  - **Math**
  - **Throwable**

java.lang.Object

- Última superclasse de todas as classes
- Define comportamento básico

```
public boolean equals(Object obj)

public native int hashCode()

public String toString()

protected void finalize() throws Throwable

protected native Object clone() throws
                    CloneNotSupportedException
```

Reescrevendo metodo toString

```
public class Pessoa {
    String nome;
    int idade;

    Pessoa(String nome, int idade) {
     this.nome = nome;
     this.idade = idade;
    }

    public static void main(String[] args) {
     Pessoa p1 = new Pessoa(”Maria”,20);
     System.out.println(p1);
    }
}
```

RESULTADO

C> java Pessoa

Pessoa@4d20a47e

Reescrevendo metodo toString

```
public class Pessoa {
    String nome;
    int idade;
    Pessoa(String nome, int idade) {
     this.nome = nome;
     this.idade = idade;
    }
    public static void main(String[] args) {
     Pessoa p1 = new Pessoa(”Maria”,20);
     System.out.println(p1);
    }
    public String toString() {
     return “NOME:”+nome+” IDADE:”+idade;
    }
}
```

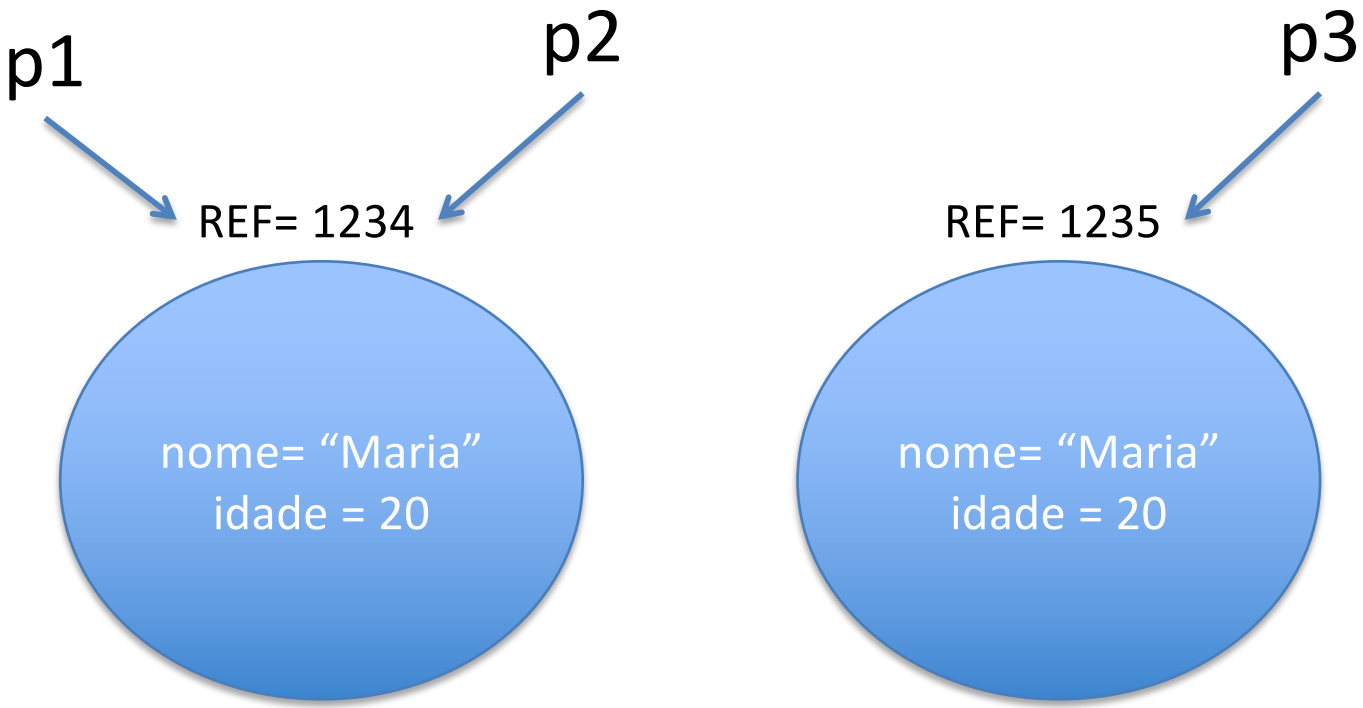
RESULTADO

C> java Pessoa

NOME:Maria IDADE:20

p1 é igual ao p2 ?
p1 é igual ao p3 ?

```
Pessoa p1 = new Pessoa("Maria",20);
Pessoa p2 = p1;
Pessoa p3 = new Pessoa("Maria",20);
```



Como comparar Objetos

- Usando a referencia do objeto:

```
if (p1 == p2) {
    System.out.println(p1+“ é igual a ”+p2)
}
```

- Usando o estado do estado:

```
if (p1.equals(p2)) {
    System.out.println(p1+“ é igual a ”+p2)
}
```

Comparando Objetos

```
public class Pessoa {
    ...
    public static void main(String[] args) {
     Pessoa p1 = new Pessoa("Maria",20);
     Pessoa p2 = p1;
     Pessoa p3 = new Pessoa("Maria",20);
     if (p1 == p2) {
         System.out.println("POR REFERENCIA "+p1+" é igual a "+p2);
     } else {
         System.out.println("POR REFERENCIA "+p1+" é diferente de "+p2);
     }
     if (p1.equals(p3)) {
         System.out.println("POR ESTADO "+p1+" é igual a "+p2);
     } else {
         System.out.println("POR ESTADO "+p1+" é diferente de "+p2);
     }
```

RESULTADO

POR REFERENCIA NOME:Maria IDADE:20 é igual a NOME:Maria IDADE:20
POR ESTADO NOME:Maria IDADE:20 é diferente de NOME:Maria IDADE:20

Altere o codigo e veja o que acontece.

```
public class Pessoa {
    ...
    public static void main(String[] args) {
     String p1 = "Maria";
     String p2 = p1;
     String p3 = "Maria";
     if (p1 == p2) {
         System.out.println("POR REFERENCIA "+p1+" é igual a "+p2);
     } else {
         System.out.println("POR REFERENCIA "+p1+" é diferente de "+p2);
     }
     if (p1.equals(p3)) {
         System.out.println("POR ESTADO "+p1+" é igual a "+p2);
     } else {
         System.out.println("POR ESTADO "+p1+" é diferente de "+p2);
     }
```

RESULTADO

POR REFERENCIA Maria é igual a Maria
POR ESTADO Maria é igual a Maria

Método equals()

- O metodo equals() deve possuir as seguintes propriedades:
- Simetrico: Para duas referencias, a e b, `a.equals(b)` se e somente se `b.equals(a)`
- Reflexivo: Para todas referencias nao nulas, a.equals(a)
- Transitividade: Se a.equals(b) e b.equals(c), então a.equals(c)
- Consistencia com hashCode(): Dois objetos iguais devem ter o mesmo hashCode()

Exemplos de Implementação de métodos equals() e hashCode()

```
class A {
    final B someNonNullField;
    C someOtherField;
    int someNonStateField;

   public boolean equals(Object other) {
       if (this == other)
          return true;
       if (!(other instanceof A))
          return false;
       A otherA = (A)other;
       return
       (someNonNullField.equals(otherA.someNonNullField) &&
        someOtherField.equals(otherA.someOtherField)&&
        someNonStateField == otherA.someNonStateField
       );
  }
...
```

Exemplos de Implementação de métodos equals() e hashCode()

```
class A {
    final B someNonNullField;
    C someOtherField;
    int someNonStateField;

    ...

    public int hashCode() {
    int hash = 1;
    hash = hash * 31 + someNonNullField.hashCode();
    hash = hash * 31
    + (someOtherField == null ? 0 : someOtherField.hashCode());
    return hash;
    }
}
```

Como fazer a copia de um objeto

- Existe um metodo chamado clone() na classe Object que pode ser reescrito para efetuar a clonagem de um objeto.
- Para usar este metodo a classe a ser clonada precisa implementar a interface Cloneable;
- Implementar o metodo clone() com uma chamada do metodo clone() da super classe:
  - super.clone();

Clonando Objetos

```
public class PessoaClone implements Cloneable {
       String nome; int idade;
       PessoaClone(String n, int i) {nome = n; idade = i;}

       public Object clone() throws CloneNotSupportedException {
          return super.clone();
       }

       public static void main(String[] args) {
                PessoaClone p1 = new PessoaClone("Maria",20);
                PessoaClone p2 = p1;
                PessoaClone p3=null;
                try {
                   p3 = (PessoaClone) p1.clone();
                } catch (CloneNotSupportedException e) {}
                System.out.println("p1:"+p1);
                System.out.println("p2:"+p2);
                System.out.println("p3:"+p3);
        }
}
```

RESULTADO

p1:PessoaClone@34aeffdf
p2:PessoaClone@34aeffdf
p3:PessoaClone@4d20a47e

Classes para Tipos Básicos

- Classes que encapsulam os tipos básicos
  - **Boolean**
  - **Byte**
  - **Character**
  - **Short**
  - **Integer**
  - **Long**
  - **Float**
  - **Double**

Exemplos de Uso

```
Boolean b = new Boolean(“true”);
if (b.booleanValue()) ... else ...;
Integer i = new Integer(“123”);
int j = i.intValue();
float f = func();
if (f==Float.POSITIVE_INFINITY ||
    f==Float.NEGATIVE_INFINITY)
  ...
else if (f==Float.NaN)
  ...
```

String & StringBuffer

- **String** = string imutável
- **StringBuffer** = string mutável

java.lang.String

- Construtores a partir de **String** e **StringBuffer**
- Construtores a partir de vetores de **byte** e **char**
- Métodos de consulta, comparação e pesquisa
- Métodos de conversão
- Métodos de *parsing*

java.lang.String

```
String texto = new String("Exemplo 1");
char c = texto.charAt(1);      // c=='x'
texto.compareTo("exemplo 1"); // < 0
char[] dst = new char[5];      // dst={0,0,0,0,0}
texto.getChars(2,5,dst,1);     // dst={0,e,m,p,0}
texto.indexOf('o',2);          // == 6
texto.lastIndexOf('w');        // == -1
texto = texto.toLowerCase();   // texto=="exemplo 1"
texto = String.valueOf(12.5); // texto=="12.5"
```

java.lang.StringBuffer

- Construtor para ajustar capacidade
- Construtor a partir de **String**
- Métodos de consulta
- Métodos de alteração

java.lang.StringBuffer

```
StringBuffer buf = new StringBuffer("Exemplo 2");
char c = buf.charAt(1);      // c=='x'
buf.capacity();              // == 25
buf.append(01).append('0'); // buf=="Exemplo 210"
buf.setCharAt(0,'e');        // buf=="exemplo 210"
buf.reverse();               // buf=="012 olpmexe"
```

java.lang.System

- Classe final
- Não pode ser instanciada
- Provê acesso ao sistema operacional
- *Streams* de entrada, saída e saída de erro
- Acesso a propriedades do sistema

java.lang.System

```
System.gc();

System.out.println("olá");

System.getProperty("user.name");

System.getProperty("java.class.path");

System.getProperty("os.name");

System.runFinalizersOnExit(true);

System.in.read();

System.exit(0);
```

java.lang.Math

- Classe final
- Não pode ser instanciada
- Provê funções matemáticas básicas

java.lang.Math

```
Math.sin(3*Math.PI/2);

Math.asin(0);

Math.exp(2);

Math.pow(Math.E, 2);

Math.sqrt(9);

Math.max(2,3);

Math.min(2,3);
```

java.lang.Math

- Faz parte da biblioteca padrão
- Várias Funções Matemáticas:
  - trigonométricas
  - logarítmicas / exponenciais
  - arredondamento
  - raizes / potências
  - números aleatórios
  - constante PI e E

java.lang.Math

- Exemplo

```
class valorPI
{
  public static void main( String p[])
  {
    System.out.println("Valor de PI "+Math.PI);
  }
}
```

Leituras

- Livro: "Java Como Programar", Deitel & Deitel
  - Capitulo 9 – Herança
- Tutorial Java (http://docs.oracle.com/javase/tutorial/)
  - Object Class:
    - http://docs.oracle.com/javase/tutorial/java/IandI/objectclass.html
  - String:
    - http://docs.oracle.com/javase/tutorial/java/data/strings.html
  - System:
    - http://docs.oracle.com/javase/tutorial/essential/environment/sysprop.html