

Programação Orientada a Objetos II

REFLEXÃO computacional

Livro de Eduardo Guerra

2024/1 - Ciência da Computação – IFC Bnu - Prof. Hylson

Material



GUERRA, Eduardo. **Componentes reutilizáveis em Java com Reflexão e anotações**. Editora Casa do código. 2014.

*Pesquisador na Universidade
livre de Bozen-Bolzano (it)*

GUERRA, 2014

1. Conhecendo a reflexão
2. Java reflection API
3. Metadados e anotações



Seção 1: conhecendo a reflexão

Exemplo de necessidade

```
@WebServlet("/novoProduto")
public class NovoProdutoServlet extends HttpServlet{
    protected void doPost(HttpServletRequest rq, HttpServletResponse rp)
        throws ServletException, IOException {
        Produto p = new Produto();
        p.setNome(rq.getParameter("nome"));
        p.setCategoria(rq.getParameter("categoria"));
        p.setPreco(Double.parseDouble(rq.getParameter("preco")));
        p.setDescricao(rq.getParameter("descricao"))
        //outras informações
    }
}
```

Exemplo de necessidade

```
@WebServlet("/cadastro")
public class CadastroClienteServlet extends HttpServlet{
    protected void doPost(HttpServletRequest rq, HttpServletResponse rp)
        throws ServletException, IOException {
        Cliente c = new Cliente();
        c.setNome(rq.getParameter("nome"));
        DateFormat formatadorData = new SimpleDateFormat("MM/dd/yy");
        c.setDataNascimento(
            (Date)formatter.parse(rq.getParameter("dataNascimento")));
        c.setLogin(rq.getParameter("login"));
        c.setSenha(rq.getParameter("senha"));
        //outras informações
    }
}
```

Lógica similar, mas
pouco reaproveitamento

Exemplo de necessidade

```
@WebServlet("/cadastro")
public class CadastroClienteServlet extends HttpServlet{
    protected void doPost(HttpServletRequest rq, HttpServletResponse rp)
        throws ServletException, IOException {
        Cliente c = new Cliente();
        c.setNome(rq.getParameter("nome"));
        DateFormat formatadorData = new SimpleDateFormat("MM/dd/yy");
        c.setDataNascimento(
            (Date)formatter.parse(rq.getParameter("dataNascimento")));
        c.setLogin(rq.getParameter("login"));
        c.setSenha(rq.getParameter("senha"));
        //outras informações
    }
}
```

Copiar e colar?



Lógica similar, mas
pouco reaproveitamento

Possível solução: “if”?

```
@WebServlet("/cadastro")
public class CadastroClienteServlet extends HttpServlet{
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        Cliente c = new Cliente()
        c.setNome(request.getParameter("nome"));
        DateFormat formatadorData = new SimpleDateFormat("MM/dd/yy");
        c.setDataNascimento(
            (Date)formatter.parse(request.getParameter("dataNascimento")));
        c.setLogin(request.getParameter("login"));
        c.setSenha(request.getParameter("senha"));
        //outras informações
    }
}
```

```
If request.getParameter("classe").equals("cliente") {
    ...new cliente...
} else if request.getParameter("classe").equals("produto")
{
    ...new produto...
}
```


Possível solução: “if”?

```
@WebServlet("/cadastro")
public class CadastroClienteServlet extends HttpServlet{
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        Cliente c = new Cliente()
        c.setNome(request.getParameter("nome"));
        DateFormat formatadorData = new SimpleDateFormat("MM/dd/yy");
        c.setDataNascimento(
            (Date)formatter.parse(request.getParameter("dataNascimento"));
        c.setLogin(request.getParameter("login"));
        c.setSenha(request.getParameter("senha"));
        //outras informações
    }
}
```

```
If request.getParameter("classe").equals("cliente") {
    ...new cliente...
} else if request.getParameter("classe").equals("produto")
{
    ...new produto...
}
```

Copiar e colar, agrupado por “if”s



Metaprogramação

“A **utilização de reflexão** também é conhecida como **metaprogramação**, pois com sua utilização um programa pode realizar computações a respeito dele próprio”.

Reflexão nas linguagens

- Java: não suporta, de forma nativa, a alteração de classes.
- Linguagens dinâmicas suportam adição de código e modificação de tipos. Exemplos: Ruby, python, Groovy.

Reflexão nas linguagens

- Java: não suporta, de forma nativa, a alteração de classes.
(linguagem compilada, tipagem estática)
- Linguagens dinâmicas suportam adição de código e modificação de tipos.
Exemplos: Ruby, python, Groovy.
(linguagens interpretadas, tipagem dinâmica)

Benefícios da reflexão

“Tornar o código mais adaptável
à estrutura dos objetos”



código reutilizável

Exemplo 1: classes em mapas

```
public class Produto {  
    private String nome;  
        private String categoria;  
        private Double preco;  
        private String descricao;  
  
        (sets, gets, toString)  
}
```

Exemplo 1: classes em mapas

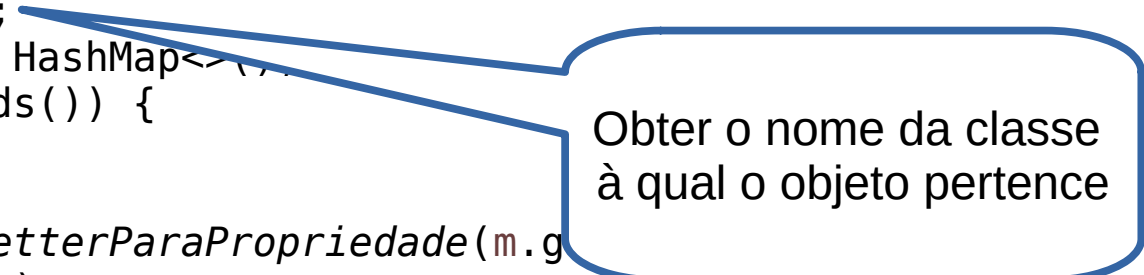
```
public class GeradorMapa {  
    public static Map<String, Object> gerarMapa(Object o) {  
        Class<?> classe = o.getClass();  
        Map<String, Object> mapa = new HashMap<>();  
        for (Method m: classe.getMethods()) {  
            try {  
                if (isGetter(m)) {  
                    String propriedade = deGetterParaPropriedade(m.getName());  
                    Object valor = m.invoke(o);  
                    mapa.put(propriedade, valor);  
                }  
            } catch (Exception e) {  
                throw new RuntimeException("Problema ao gerar o mapa");  
            }  
        }  
        return mapa;  
    }  
}
```

Recebe como parâmetro
um objeto

Retorna um **mapa** de
strings que são chaves,
cujos valores podem
ser qualquer tipo
de objeto

Exemplo 1: classes em mapas

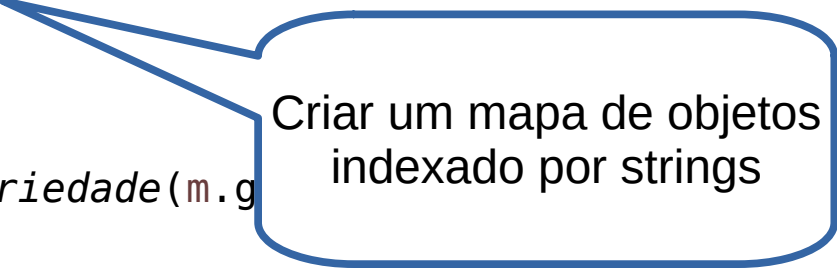
```
public class GeradorMapa {  
    public static Map<String, Object> gerarMapa(Object o) {  
        Class<?> classe = o.getClass();  
        Map<String, Object> mapa = new HashMap<>();  
        for (Method m: classe.getMethods()) {  
            try {  
                if (isGetter(m)) {  
                    String propriedade = deGetterParaPropriedade(m.getName());  
                    Object valor = m.invoke(o);  
                    mapa.put(propriedade, valor);  
                }  
            } catch (Exception e) {  
                throw new RuntimeException("Problema ao gerar o mapa", e);  
            }  
        }  
        return mapa;  
    }  
}
```



Obter o nome da classe
à qual o objeto pertence

Exemplo 1: classes em mapas

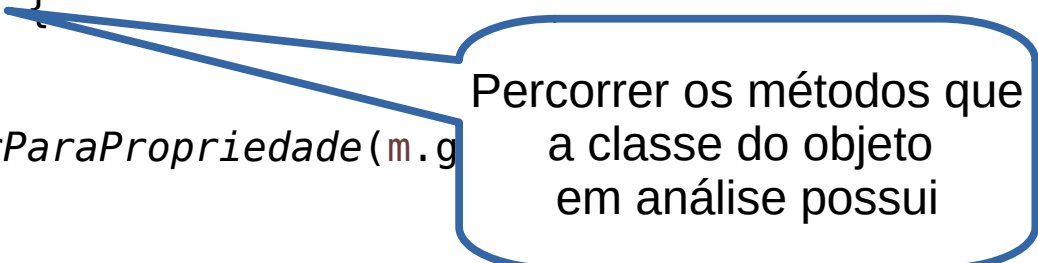
```
public class GeradorMapa {  
    public static Map<String, Object> gerarMapa(Object o) {  
        Class<?> classe = o.getClass();  
        Map<String, Object> mapa = new HashMap<>();  
        for (Method m: classe.getMethods()) {  
            try {  
                if (isGetter(m)) {  
                    String propriedade = deGetterParaPropriedade(m.getName());  
                    Object valor = m.invoke(o);  
                    mapa.put(propriedade, valor);  
                }  
            } catch (Exception e) {  
                throw new RuntimeException("Problema ao gerar o mapa", e);  
            }  
        }  
        return mapa;  
    }  
}
```



Criar um mapa de objetos indexado por strings

Exemplo 1: classes em mapas

```
public class GeradorMapa {  
    public static Map<String, Object> gerarMapa(Object o) {  
        Class<?> classe = o.getClass();  
        Map<String, Object> mapa = new HashMap<>();  
        for (Method m: classe.getMethods()) {  
            try {  
                if (isGetter(m)) {  
                    String propriedade = deGetterParaPropriedade(m.getName());  
                    Object valor = m.invoke(o);  
                    mapa.put(propriedade, valor);  
                }  
            } catch (Exception e) {  
                throw new RuntimeException("Problema ao gerar o mapa", e);  
            }  
        }  
        return mapa;  
    }  
}
```



Percorrer os métodos que a classe do objeto em análise possui

Exemplo 1: classes em mapas

```
public class GeradorMapa {  
    public static Map<String, Object> gerarMapa(Object o) {  
        Class<?> classe = o.getClass();  
        Map<String, Object> mapa = new HashMap<>();  
        for (Method m: classe.getMethods()) {  
            try {  
                if (isGetter(m)) {  
                    String propriedade = deGetterParaPropriedade(m.getName());  
                    Object valor = m.invoke(o);  
                    mapa.put(propriedade, valor);  
                }  
            } catch (Exception e) {  
                throw new RuntimeException("Problema ao gerar o mapa", e);  
            }  
        }  
        return mapa;  
    }  
}
```

Esse método é um “get”?
(função descrita em breve)

Exemplo 1: classes em mapas

```
public class GeradorMapa {  
    public static Map<String, Object> gerarMapa(Object o) {  
        Class<?> classe = o.getClass();  
        Map<String, Object> mapa = new HashMap<>();  
        for (Method m: classe.getMethods()) {  
            try {  
                if (isGetter(m)) {  
                    String propriedade = deGetterParaPropriedade(m.getName());  
                    Object valor = m.invoke(o);  
                    mapa.put(propriedade, valor);  
                }  
            } catch (Exception e) {  
                throw new RuntimeException("Problema ao gerar o mapa", e);  
            }  
        }  
        return mapa;  
    }  
}
```

Obtém o nome do atributo
referente ao método get
(função descrita em breve)

Exemplo 1: classes em mapas

```
public class GeradorMapa {  
    public static Map<String, Object> gerarMapa(Object o) {  
        Class<?> classe = o.getClass();  
        Map<String, Object> mapa = new HashMap<>();  
        for (Method m: classe.getMethods()) {  
            try {  
                if (isGetter(m)) {  
                    String propriedade = deGetterParaPropriedade(m.getName());  
                    Object valor = m.invoke(o);  
                    mapa.put(propriedade, valor);  
                }  
            } catch (Exception e) {  
                throw new RuntimeException("Problema ao gerar o mapa");  
            }  
        }  
        return mapa;  
    }  
}
```

Executa o método (m) no objeto (o) e obtém seu valor de retorno (de qualquer tipo)

Exemplo 1: classes em mapas

```
public class GeradorMapa {  
    public static Map<String, Object> gerarMapa(Object o) {  
        Class<?> classe = o.getClass();  
        Map<String, Object> mapa = new HashMap<>();  
        for (Method m: classe.getMethods()) {  
            try {  
                if (isGetter(m)) {  
                    String propriedade = deGetterParaPropriedade(m.getName());  
                    Object valor = m.invoke(o);  
                    mapa.put(propriedade, valor);  
                }  
            } catch (Exception e) {  
                throw new RuntimeException("Problema ao gerar o mapa");  
            }  
        }  
        return mapa;  
    }  
}
```

Armazena o nome do atributo e o valor do atributo no mapa

O mapa é retornado ao final do processamento

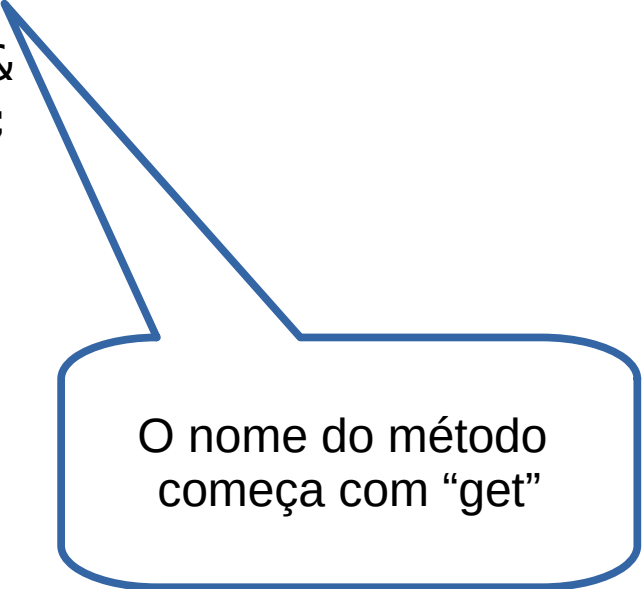
Exemplo 1: classes em mapas

Como saber se um
método é “get”?

Exemplo 1: classes em mapas

```
private static boolean isGetter(Method m) {  
    return m.getName().startsWith("get") &&  
           m.getReturnType() != void.class &&  
           m.getParameterTypes().length == 0;  
}
```

Como saber se um
método é “get”?

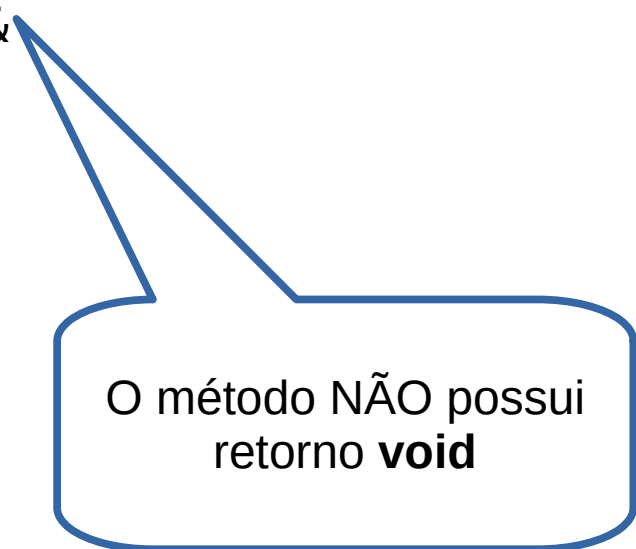


O nome do método
começa com “get”

Exemplo 1: classes em mapas

```
private static boolean isGetter(Method m) {  
    return m.getName().startsWith("get") &&  
           m.getReturnType() != void.class &&  
           m.getParameterTypes().length == 0;  
}
```

Como saber se um
método é “get”?



O método NÃO possui
retorno **void**

Exemplo 1: classes em mapas

```
private static boolean isGetter(Method m) {  
    return m.getName().startsWith("get") &&  
           m.getReturnType() != void.class &&  
           m.getParameterTypes().length == 0;  
}
```

Como saber se um
método é “get”?

Vetor de tipos dos
parâmetros; se não
houver parâmetros,
tamanho do vetor = 0

Exemplo 1: classes em mapas

Como obter o nome do atributo retornado pelo get?

*Exemplo: ~~get~~**Preco** => **preco***

Exemplo 1: classes em mapas

```
private static String deGetterParaPropriedade(String nomeGetter) {  
    StringBuffer retorno = new StringBuffer();  
    retorno.append(nomeGetter.substring(3, 4).toLowerCase());  
    retorno.append(nomeGetter.substring(4));  
    return retorno.toString();  
}
```

Como obter o nome do atributo retornado pelo get?



Criar uma string auxiliar

*Exemplo: ~~get~~**Preco** => **preco***

Exemplo 1: classes em mapas

```
private static String deGetterParaPropriedade(String nomeGetter) {  
    StringBuffer retorno = new StringBuffer();  
    retorno.append(nomeGetter.substring(3, 4).toLowerCase());  
    retorno.append(nomeGetter.substring(4));  
    return retorno.toString();  
}
```

Como obter o nome do atributo retornado pelo get?

*Exemplo: get**P**reco => **p**reco*

[g, e, t, **P**, r, e, c, o]
[0, 1, 2, 3, **4**, 5, 6, 7]

Obter o primeiro caracter depois do "get" (posição 3), em minúsculo

Copia caracter da posição 3 até a posição 4 (mas "exclui" o caracter da posição 4)

Exemplo 1: classes em mapas

```
private static String deGetterParaPropriedade(String nomeGetter) {  
    StringBuffer retorno = new StringBuffer();  
    retorno.append(nomeGetter.substring(3, 4).toLowerCase());  
    retorno.append(nomeGetter.substring(4));  
    return retorno.toString();  
}
```

Como obter o nome do atributo retornado pelo get?

Obter o “restante” do nome do método

Exemplo: ~~get~~**P**reco => **p**reco

Copia caracter da posição 4 em diante

[g, e, t, P, r, e, c, o]
[0, 1, 2, 3, 4, 5, 6, 7]

Exemplo 1: classes em mapas

```
private static String deGetterParaPropriedade(String nomeGetter) {  
    StringBuffer retorno = new StringBuffer();  
    retorno.append(nomeGetter.substring(3, 4).toLowerCase());  
    retorno.append(nomeGetter.substring(4));  
    return retorno.toString();  
}
```

Como obter o nome do atributo retornado pelo get?

Retornar as duas partes extraídas (substring) e reunidas (append) na variável auxiliar

Exemplo: ~~get~~**Preco** => **preco**

Teste do mapa com produto

```
Produto p = new Produto();
```

```
p.setNome("Caderno");
```

```
p.setCategoria("Material escolar");
```

```
p.setPreco(13.00);
```

```
p.setDescricao("Caderno pauta dupla "  
+ "100 páginas tilibra");
```

Criação do produto
e definição de valores

```
System.out.println(p); //invocação método toString
```

```
Map<String, Object> props = GeradorMapa.gerarMapa(p);
```

```
for (String prop : props.keySet()) {
```

```
    System.out.println(prop + " = " + props.get(prop));
```

```
}
```


Teste do mapa com produto

```
Produto p = new Produto();
```

```
p.setNome("Caderno");  
p.setCategoria("Material escolar");  
p.setPreco(13.00);  
p.setDescricao("Caderno pauta dupla "  
    + "100 páginas tilibra");
```

```
System.out.println(p); //invocação método toString
```

```
Map<String, Object> props = GeradorMapa.gerarMapa(p);  
for (String prop : props.keySet()) {  
    System.out.println(prop + " = " + props.get(prop));  
}
```

Exibição do produto
via método toString

```
Produto [nome=Caderno, categoria=Material escolar, preco=13.0, descricao=Caderno pauta dupla 100 páginas tilibra]
```

Teste do mapa com produto

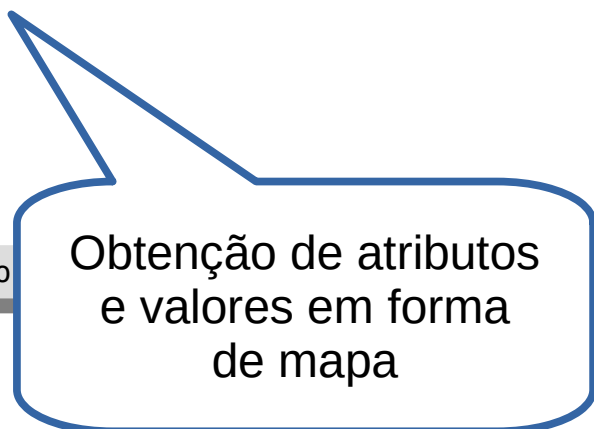
```
Produto p = new Produto();
```

```
p.setNome("Caderno");  
p.setCategoria("Material escolar");  
p.setPreco(13.00);  
p.setDescricao("Caderno pauta dupla "  
    + "100 páginas tilibra");
```

```
System.out.println(p); //invocação método toString
```

```
Map<String, Object> props = GeradorMapa.gerarMapa(p);  
for (String prop : props.keySet()) {  
    System.out.println(prop + " = " + props.get(prop));  
}
```

```
Produto [nome=Caderno, categoria=Material escolar, preco=13.0, descricao=Caderno
```



Obtenção de atributos
e valores em forma
de mapa

Teste do mapa com produto

```
Produto p = new Produto();
```

```
p.setNome("Caderno");
```

```
p.setCategoria("Material escolar");
```

```
p.setPreco(13.00);
```

```
p.setDescricao("Caderno pauta dupla  
+ "100 páginas tilibra");
```

```
preco = 13.0
```

```
categoria = Material escolar
```

```
nome = Caderno
```

```
class = class modelo.Produto
```

```
descricao = Caderno pauta dupla 100 páginas tilibra
```

```
System.out.println(p); //invocação método toString
```

```
Map<String, Object> props = GeradorMapa.gerarMapa(p);
```

```
for (String prop : props.keySet()) {
```

```
    System.out.println(prop + " = " + props.get(prop));
```

```
}
```

Exibindo nomes dos
atributos e valores

```
Produto [nome=Caderno, categoria=Material escolar, preco=13.0, descricao=Caderno pauta dupla 100 páginas tilibra]
```

Anotações

“Permitem marcar os elementos de forma que um algoritmo que utilize reflexão possa identificar os elementos que ele deve tratar de forma diferente”

Exemplo de anotações

```
@Retention(RetentionPolicy.RUNTIME)  
public @interface Ignorar {  
}
```

Anotação personalizada
chamada **Ignorar**

```
@Retention(RetentionPolicy.RUNTIME)  
public @interface NomePropriedade {  
    String value();  
}
```

Anotação personalizada chamada
NomePropriedade; possui
valor de retorno

Exemplo de anotações

```
public class Telefone {  
    private String numero;  
    private String codigoPais;  
    private String operadora;  
  
    public String getNumero() { return numero; }  
    public void setNumero(String n) { numero = n; }
```

Exemplo de anotações

```
// anotação para permitir acesso ao campo de outra forma  
// similar a um "alias" em tempo de execução apenas  
@NomePropriedade("codigoInternacional")  
public String getCodigoPais() {  
    return codigoPais;  
}
```

Uso da anotação
Personalizada.

```
// anotação para ignorar este método  
@Ignorar  
public String getOperadora() { return operadora; }
```

Uso da anotação
Personalizada.

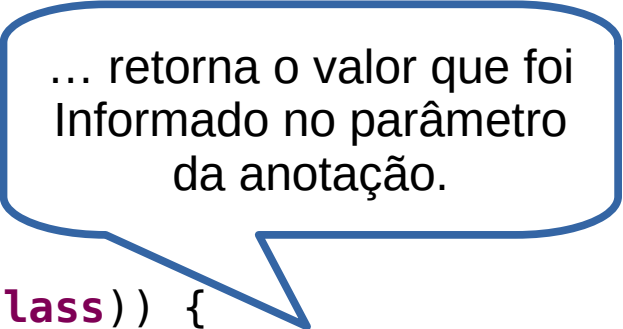
Exemplo de anotações

```
public static Map<String, Object> gerarMapaComAnotacao(Object o) {  
    Class<?> classe = o.getClass();  
    Map<String, Object> mapa = new HashMap<>();  
    for (Method m: classe.getMethods()) {  
        try {  
            if (isGetter(m)) {  
                String propriedade = null;  
                if (m.isAnnotationPresent(NomePropriedade.class)) {  
                    propriedade = m.getAnnotation(NomePropriedade.class).value();  
                } else {  
                    propriedade = deGetterParaPropriedade(m.getName());  
                }  
  
                Object valor = m.invoke(o);  
                mapa.put(propriedade, valor);  
            }  
        }  
    }  
    ...  
}
```

Se o método get estiver
com uma certa
anotação específica...

Exemplo de anotações

```
public static Map<String, Object> gerarMapaComAnotacao(Object o) {  
    Class<?> classe = o.getClass();  
    Map<String, Object> mapa = new HashMap<>();  
    for (Method m: classe.getMethods()) {  
        try {  
            if (isGetter(m)) {  
                String propriedade = null;  
                if (m.isAnnotationPresent(NomePropriedade.class)) {  
                    propriedade = m.getAnnotation(NomePropriedade.class).value();  
                } else {  
                    propriedade = deGetterParaPropriedade(m.getName());  
                }  
  
                Object valor = m.invoke(o);  
                mapa.put(propriedade, valor);  
            }  
        }  
    }  
    ...  
}
```



... retorna o valor que foi
Informado no parâmetro
da anotação.

Exemplo de anotações

```
public static Map<String, Object> gerarMapaComAnotacao(Object o) {  
    Class<?> classe = o.getClass();  
    Map<String, Object> mapa = new HashMap<>();  
    for (Method m: classe.getMethods()) {  
        try {  
            if (isGetter(m)) {  
                String propriedade = null;  
                if (m.isAnnotationPresent(NomePropriedade.class)) {  
                    propriedade = m.getAnnotation(NomePropriedade.class).value();  
                } else {  
                    propriedade = deGetterParaPropriedade(m.getName());  
                }  
  
                Object valor = m.invoke(o);  
                mapa.put(propriedade, valor);  
            }  
        }  
    }  
    ...  
}
```

Caso contrário, atua como no método original: pega o valor do atributo a partir do nome do método

Exemplo de anotações

```
private static boolean isGetter(Method m) {  
    return m.getName().startsWith("get") &&  
        m.getReturnType() != void.class &&  
        m.getParameterTypes().length == 0 &&  
        !m.isAnnotationPresent(Ignorar.class);  
}
```

O verificador de método “get” também considera se o método não possui a anotação “Ignorar”.

Exemplo de anotações

```
Telefone t = new Telefone();

t.setNumero("47 9912 1314");
t.setOperadora("VIVO");
t.setCodigoPais("55");

// exibição via toString
System.out.println("Exibição via toString:");
System.out.println(t);

// exibição via reflexão
System.out.println("\nExibição com reflexão:");
Map<String, Object> props = GeradorMapaComAnotacao.gerarMapaComAnotacao(t);
for (String prop : props.keySet()) {
    System.out.println(prop + " = " + props.get(prop));
}

// exibição normal
System.out.println("\nExibição com gets:");
System.out.println("Número: " + t.getNumero());
System.out.println("Operadora: " + t.getOperadora());
System.out.println("Codigo do país: " + t.getCodigoPais());
```

Exemplo de anotações

```
Telefone t = new Telefone();
```

```
t.setNumero("47 9912 1314");  
t.setOperadora("VIVO");  
t.setCodigoPais("55");
```

Exibição via toString:
Telefone [numero=47 9912 1314, codigoPais=55, operadora=VIVO]

```
// exibição via toString  
System.out.println("Exibição via toString:");  
System.out.println(t);  
  
// exibição via reflexão  
System.out.println("\nExibição com reflexão:");  
Map<String, Object> props = GeradorMapaComAnotacao.gerarMapaComAnotacao(t);  
for (String prop : props.keySet()) {  
    System.out.println(prop + " = " + props.get(prop));  
}
```

Na exibição de informações via método toString, todos os atributos estáticos são listados

```
// exibição normal  
System.out.println("\nExibição com gets:");  
System.out.println("Número: " + t.getNumero());  
System.out.println("Operadora: " + t.getOperadora());  
System.out.println("Codigo do país: " + t.getCodigoPais());
```

Exemplo de anotações

```
Telefone t = new Telefone();  
t.setNumero("47 9912 1314");  
t.setOperadora("VIVO");  
t.setCodigoPais("55");
```

```
// exibição via toString  
System.out.println("Exibição via toString:");  
System.out.println(t);  
  
// exibição via reflexão  
System.out.println("\nExibição com reflexão:");  
Map<String, Object> props = GeradorMapaComAnotacao.gerarMapaComAnotacao(t);  
for (String prop : props.keySet()) {  
    System.out.println(prop + " = " + props.get(prop));  
}
```

```
// exibição normal  
System.out.println("\nExibição com gets:");  
System.out.println("Número: " + t.getNumero());  
System.out.println("Operadora: " + t.getOperadora());  
System.out.println("Codigo do país: " + t.getCodigoPais());
```

```
@NomePropriedade("codigoInternacional")  
public String getCodigoPais() {  
    return codigoPais;  
}
```

Exibição com reflexão:
codigoInternacional = 55
numero = 47 9912 1314
class = class modelo.Telefone

A exibição com uso de reflexão mostra o atributo “anotado”, bem como ignora o atributo “**operadora**”

Exemplo de anotações

```
Telefone t = new Telefone();

t.setNumero("47 9912 1314");
t.setOperadora("VIVO");
t.setCodigoPais("55");

// exibição via toString
System.out.println("Exibição via toString:");
System.out.println(t);

// exibição via reflexão
System.out.println("\nExibição com reflexão:");
Map<String, Object> props = GeradorMapaComAnotacao.gerarMapaComAnotacao(t);
for (String prop : props.keySet()) {
    System.out.println(prop + " = " + props.get(prop));
}

// exibição normal
System.out.println("\nExibição com gets:");
System.out.println("Número: " + t.getNumero());
System.out.println("Operadora: " + t.getOperadora());
System.out.println("Codigo do país: " + t.getCodigoPais());
```

Exibição com gets:
Número: 47 9912 1314
Operadora: VIVO
Codigo do país: 55

Usar os métodos get na exibição apresenta os valores esperados.

Capítulo 2: Java reflection API

Conceitos

Uma classe possui metainformações
a respeito de seus objetos

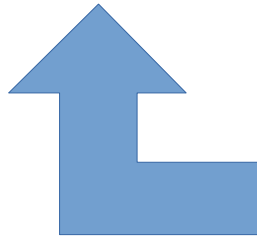
Conceitos

Uma classe possui metainformações
a respeito de seus objetos

*A classe descreve o objeto, mas
quem descreve a classe?*

Conceitos

Uma instância da classe **Class**
descreve as informações de uma classe



A classe descreve o objeto, mas
quem descreve a classe?

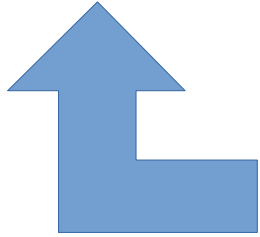
Conceitos

Uma instância da classe **Class**
descreve as informações de uma classe

*Quem descreve a instância de **Class**?*

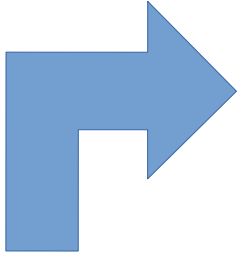
Conceitos

A classe **Class**!



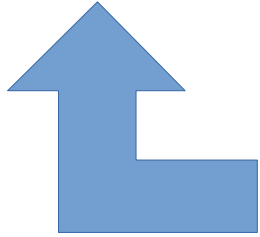
*Quem descreve a instância de **Class**?*

Conceitos



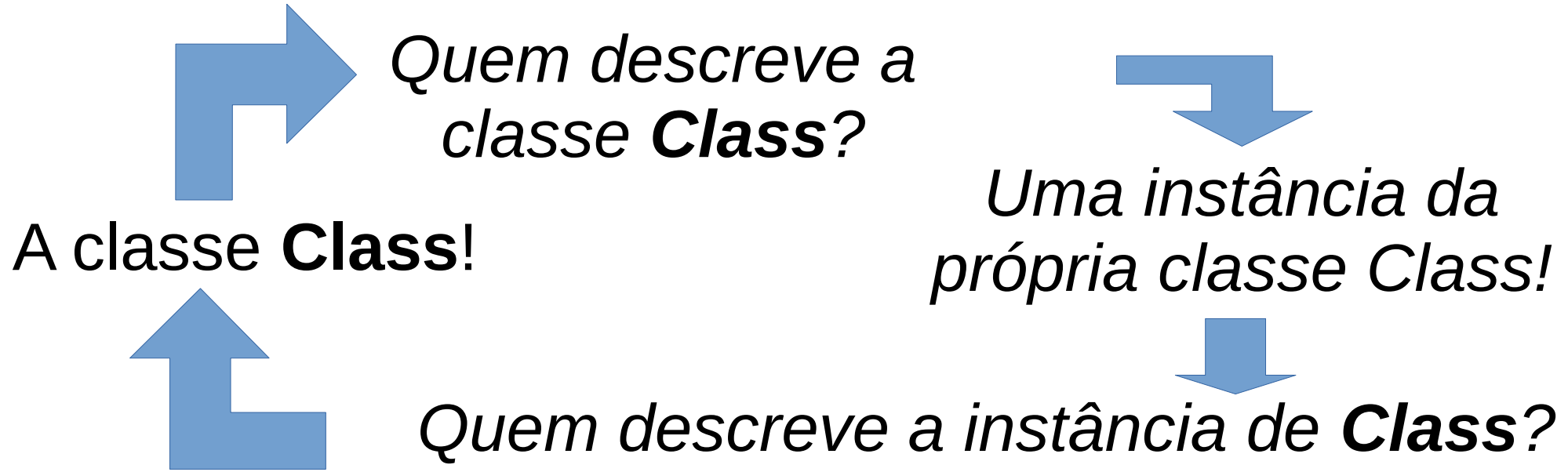
*Quem descreve a
classe **Class**?*

A classe **Class**!

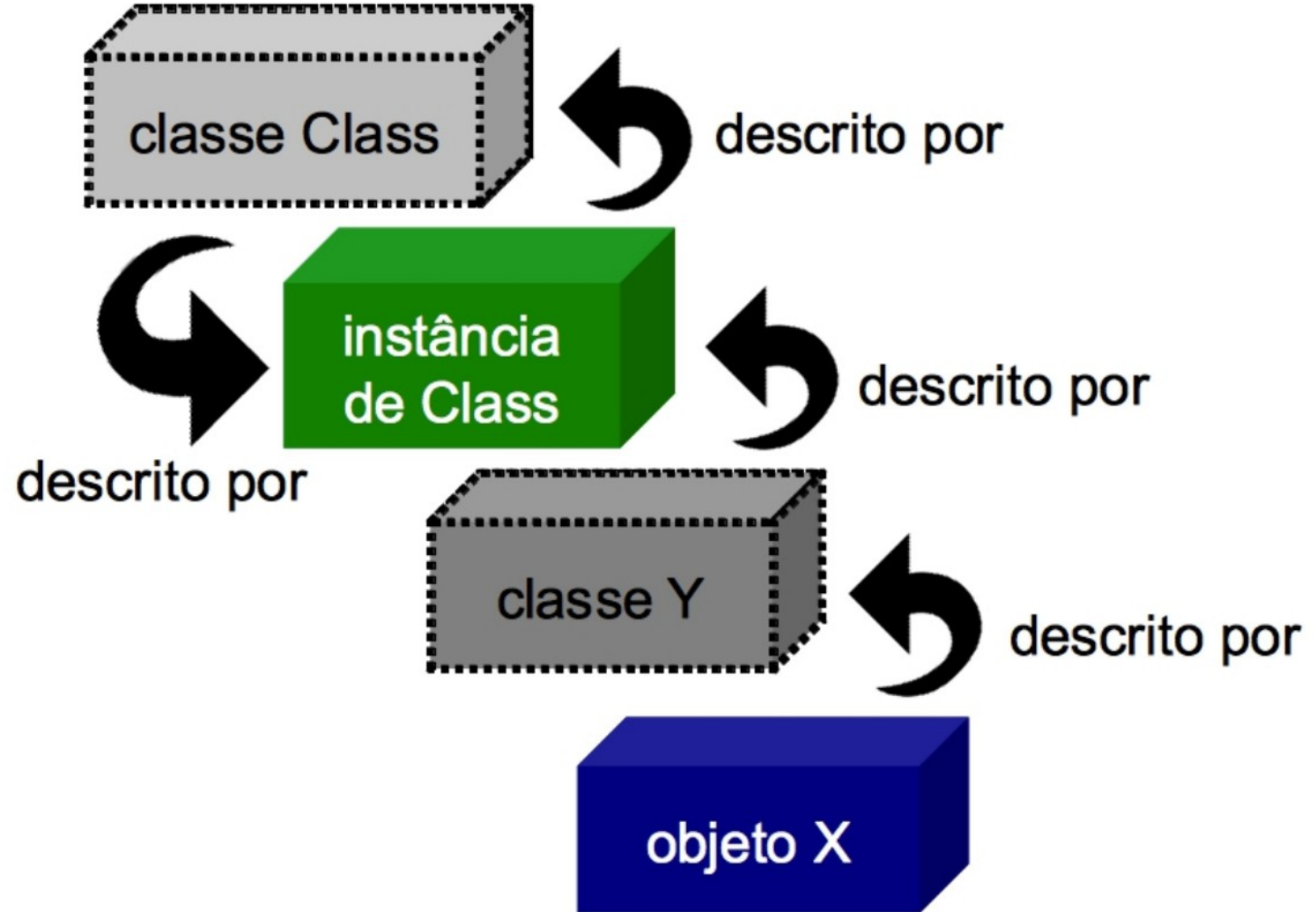


*Quem descreve a instância de **Class**?*

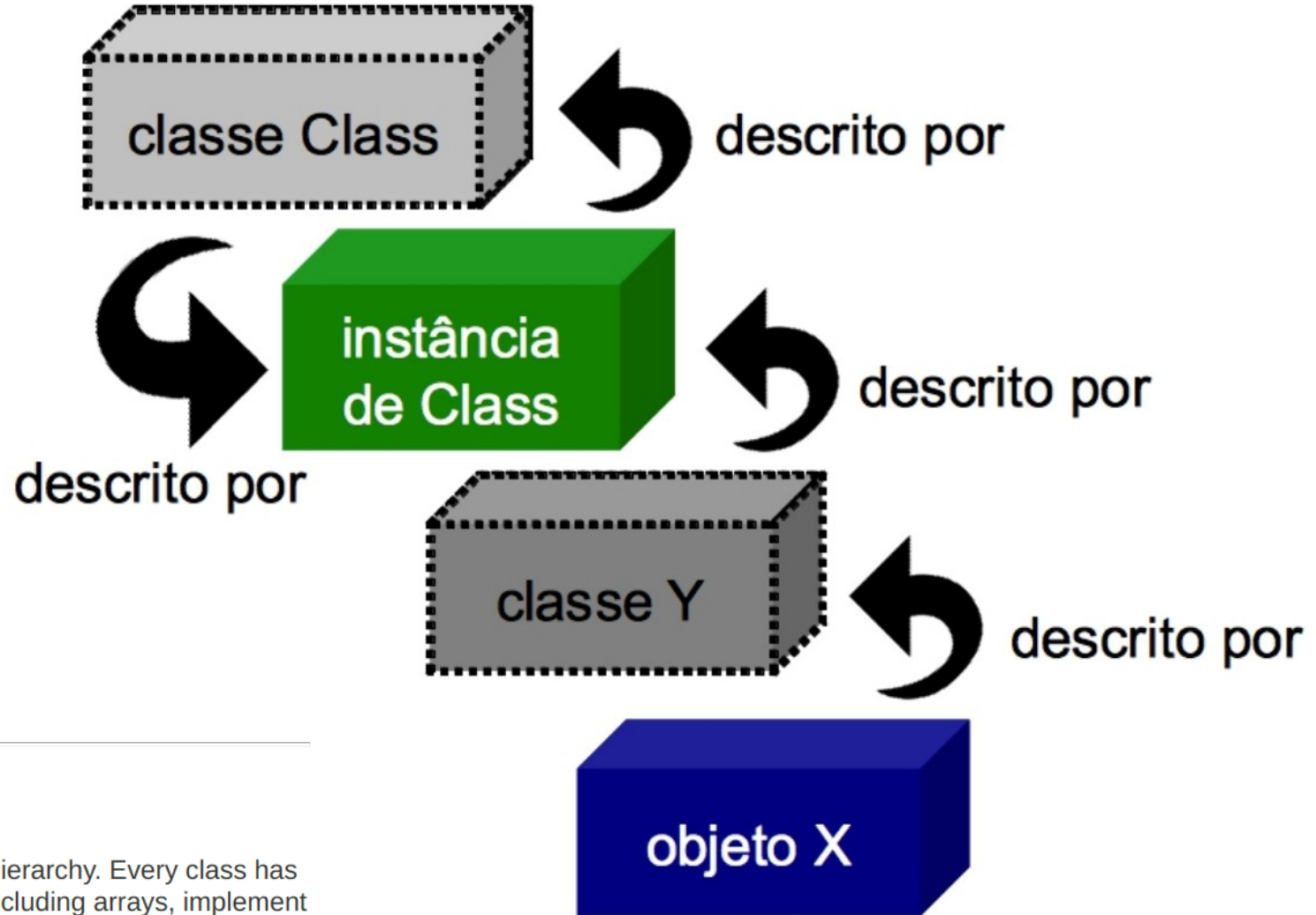
Conceitos



Conceitos



Conceitos



java.lang

Class Object

java.lang.Object

```
public class Object
```

Class **Object** is the root of the class hierarchy. Every class has **Object** as a superclass. All objects, including arrays, implement the methods of this class.

Como obter a classe de um objeto?

Como obter a classe de um objeto?

Forma 1: via referência estática

Classe via referência estática

```
public static void main(String args[]) {  
    Class<String> classe = String.class;  
    System.out.println(classe.getName());  
    imprimeNomeClasse(Integer.class);  
    Class inst = Boolean.class;  
    System.out.println(inst.getName());  
}  
  
public static void imprimeNomeClasse(Class<?> classe) {  
    System.out.println("Chamando o método com "  
    + classe.getName());  
}
```

Uso do atributo .class

Classe via referência estática

```
public static void main(String args[]) {  
    Class<String> classe = String.class;  
    System.out.println(classe.getName());  
    imprimeNomeClasse(Integer.class);  
    Class inst = Boolean.class;  
    System.out.println(inst.getName());  
}
```

```
public static void imprimeNomeClass  
    System.out.println("Chamando o método  
    + classe.getName());  
}
```

Obter uma instância da classe.
O tipo da instância é especificado.

Uso do atributo .class

Classe via referência estática

```
public static void main(String args[]) {  
    Class<String> classe = String.class;  
    System.out.println(classe.getName());  
    imprimeNomeClasse(Integer.class);  
    Class inst = Boolean.class;  
    System.out.println(inst.getName());  
}
```

A instância da classe é
do tipo String

```
public static void imprimeNomeClasse(Class<?> classe) {  
    System.out.println("Chamando o método com "  
    + classe.getName());  
}
```

java.lang.String

Chamando o método com java.lang.Integer
java.lang.Boolean

Uso do atributo .class

Classe via referência estática

```
public static void main(String args[]) {  
    Class<String> classe = String.class;  
    System.out.println(classe.getName());  
    imprimeNomeClasse(Integer.class);  
    Class inst = Boolean.class;  
    System.out.println(inst.getName());  
}
```

A instância da classe Integer
é do tipo Integer :-)

```
public static void imprimeNomeClasse(Class<?> classe) {  
    System.out.println("Chamando o método com "  
    + classe.getName());  
}
```

```
java.lang.String  
Chamando o método com java.lang.Integer  
java.lang.Boolean
```

Uso do atributo .class

Classe via referência estática

```
public static void main(String args[]) {  
    Class<String> classe = String.class;  
    System.out.println(classe.getName());  
    imprimeNomeClasse(Integer.class);  
    Class inst = Boolean.class;  
    System.out.println(inst.getName());  
}
```

Instância de Boolean é Boolean. É recomendado, mas não obrigatório, especificar o tipo do dado esperado na atribuição.

```
public static void imprimeNomeClasse(Class<?> classe) {  
    System.out.println("Chamando o método com "  
    + classe.getName());  
}
```

```
java.lang.String  
Chamando o método com java.lang.Integer  
java.lang.Boolean
```

Uso do atributo .class

Classe via referência estática

```
public static void main(String args[]) {  
    Class<String> classe = String.class;  
    System.out.println(classe.getName());  
    imprimeNomeClasse(Integer.class);  
    Class inst = Boolean.class;  
    System.out.println(inst.getName());  
}
```

```
public static void imprimeNomeClasse(Class<?> classe) {  
    System.out.println("Chamando o método com "  
    + classe.getName());  
}
```

```
java.lang.String  
Chamando o método com java.lang.Integer  
java.lang.Boolean
```

Um método auxiliar define o parâmetro com o tipo genérico.

Uso do atributo .class

Como obter a classe de um objeto?

Forma 1: via referência estática

Forma 2: via objeto

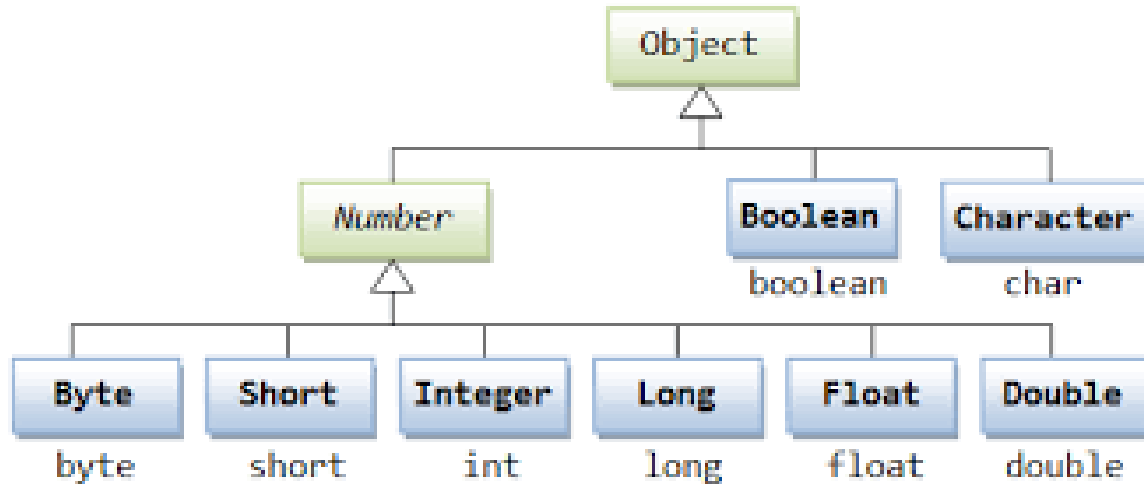
Classe via objeto

```
Number object = new Integer(100);  
Class <? extends Number> c = object.getClass();  
System.out.println(c.getName());  
System.out.println(object);  
  
Integer i = new Integer(100);  
Class <?> ni = object.getClass();  
System.out.println(ni.getName());  
System.out.println(i);
```

Uso do método getClass()

Classe via objeto

```
Number object = new Integer(100);  
object.getClass();
```



Criação de um inteiro de valor 100, armazenado em objeto hierárquico superior do tipo Number.

<https://www3.ntu.edu.sg/home/ehchua/programming/java/JavaGeneric.html>

Uso do método getClass()

Classe via objeto

```
Number object = new Integer(100);  
Class <? extends Number> c = object.getClass();  
System.out.println(c.getName());  
System.out.println(object);  
  
Integer i = new Integer(100);  
Class <?> ni = object.getClass();  
System.out.println(ni.getName());  
System.out.println(i);
```

São exibidos o nome da classe do objeto e o valor do objeto.

```
java.lang.Integer  
100
```

Uso do método getClass()

Classe via objeto

```
Number object = new Integer(100);  
Class <? extends Number> c = object.getClass();  
System.out.println(c.getName());  
System.out.println(object);  
  
Integer i = new Integer(100);  
Class <?> ni = object.getClass();  
System.out.println(ni.getName());  
System.out.println(i);
```

Resultado idêntico é alcançado sem uso da classe superior.

```
java.lang.Integer  
100  
java.lang.Integer  
100
```

Uso do método getClass()

Como obter a classe de um objeto?

Forma 1: via referência estática

Forma 2: via objeto

Forma 3: via String

Classe via String

```
Class<?> hm = Class.forName("java.util.HashMap");
System.out.println("Nome da classe: " + hm.getName());
for (Constructor c : hm.getConstructors()) {
    System.out.print("Construtor com " +
        c.getParameterCount() + " parâmetros");
    if (c.getParameterCount() > 0) {
        System.out.print(": ");
        for (Type t : c.getGenericParameterTypes()) {
            System.out.print(" (" + t.getTypeName() + ")");
        }
    }
    System.out.println(" ");
}
```

Uso do método `forName`

Classe via String

O método retorna a classe conforme a string de busca.

```
Class<?> hm = Class.forName("java.util.HashMap");
System.out.println("Nome da classe: " + hm.getName());
for (Constructor c : hm.getConstructors()) {
    System.out.print("Construtor com " +
        c.getParameterCount() + " parâmetros");
    if (c.getParameterCount() > 0) {
        System.out.print(": ");
        for (Type t : c.getGenericParameterTypes()) {
```

```
Nome da classe: java.util.HashMap
Construtor com 2 parâmetros: (int) (float)
Construtor com 0 parâmetros
Construtor com 1 parâmetros: (java.util.Map<? extends K, ? extends V>)
Construtor com 1 parâmetros: (int)
```

Uso do método forName

Classe via String

```
Class<?> hm = Class.forName("java.util.HashMap");
System.out.println("Nome da classe: " + hm.getName());
for (Constructor c : hm.getConstructors()) {
    System.out.print("Construtor com " +
        c.getParameterCount() + " parâmetros");
    if (c.getParameterCount() > 0) {
        System.out.print(": ");
        for (Type t : c.getGenericParameterTypes()) {
            System.out.print(" (" + t.getTypeName() + ")");
        }
    }
    System.out.println(" ");
}
```

Nesse exemplo, são percorridos os construtores existentes na classe encontrada.

Uso do método `forName`

Classe via String

```
Class<?> hm = Class.forName("java.util.HashMap");
System.out.println("Nome da classe: " + hm.getName());
for (Constructor c : hm.getConstructors()) {
    System.out.print("Construtor com " +
        c.getParameterCount() + " parâmetros");
    if (c.getParameterCount() > 0) {
        System.out.print(": ");
        for (Type t : c.getGenericParameterTypes()) {
            System.out.print("(" + t.getTypeName() + ")");
        }
    }
}
```

Para construtores com parâmetros, são exibidos os tipos de parâmetros.

```
Nome da classe: java.util.HashMap
} Construtor com 2 parâmetros: (int) (float)
Construtor com 0 parâmetros
Construtor com 1 parâmetros: (java.util.Map<? extends K, ? extends V>)
Construtor com 1 parâmetros: (int)
```

Classe via String

```
Class<?> hm = Class.forName("java.util.HashMap");
System.out.println("Nome da classe: " + hm.getName());
for (Constructor c : hm.getConstructors()) {
    System.out.print("Construtor com " +
        c.getParameterCount() + " parâmetros");
    if (c.getParameterCount() > 0) {
        System.out.print(": ");
        for (Type t : c.getGenericParameterTypes()) {
            System.out.print(" (" + t.getTypeName() + ")");
        }
    }
}
```

Informações sobre
parâmetros genéricos
São retornadas pelo
uso do **Generic**

```
Nome da classe: java.util.HashMap
} Construtor com 2 parâmetros: (int) (float)
Construtor com 0 parâmetros
Construtor com 1 parâmetros: (java.util.Map<? extends K, ? extends V>)
Construtor com 1 parâmetros: (int)
```

Exemplo: hierarquia de classes

Exemplo: hierarquia de classes

```
String buscar = "java.util.TreeMap";  
try {  
    Class<?> c = Class.forName(buscar);  
    imprimirHierarquia(c, 1);  
} catch (ClassNotFoundException e) {  
    System.out.println("Classe não encontrada: "+e.getMessage());  
}
```

Exemplo: hierarquia de classes

```
String buscar = "java.util.TreeMap";  
try {  
    Class<?> c = Class.forName(buscar);  
    imprimirHierarquia(c, 1);  
} catch (ClassNotFoundException e) {  
    System.out.println("Classe não encontrada: "+e.getMessage());  
}
```

Busca-se a classe TreeMap
com o método forName

Exemplo: hierarquia de classes

```
String buscar = "java.util.TreeMap";  
try {  
    Class<?> c = Class.forName(buscar);  
    imprimirHierarquia(c, 1);  
} catch (ClassNotFoundException e) {  
    System.out.println("Classe não encontrada: "+e.getMessage());  
}
```

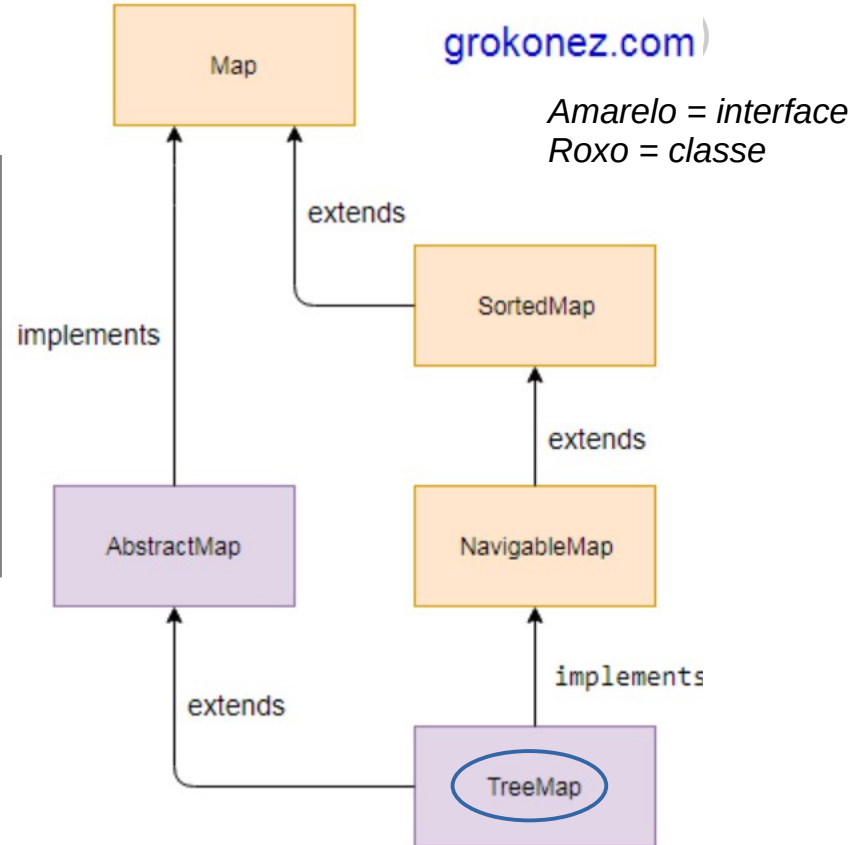
Chamamos uma função para
exibir a hierarquia de classes
e interfaces

Exemplo: hierarquia de classes

```
private static void imprimirHierarqu  
{  
    List<Class> classes = getClasses79
```

```
| -> java.util.AbstractMap (superclass)  
    | -> java.lang.Object (superclass)  
    | -> java.util.Map  
| -> java.util.NavigableMap  
    | -> java.util.SortedMap  
        | -> java.util.Map  
| -> java.lang.Cloneable  
| -> java.io.Serializable
```

```
System.out.println(" ");  
if (clazz != Object.class) {  
    imprimirHierarquia(clazz, nive  
}  
}  
}
```



Exemplo: hierarquia de classes

```
private static void imprimirHierarquia(Class clazz) {
```

```
    List<Class<?>> lista = getSuperClasses(clazz);
    for (Class<?> super : lista) {
        System.out.println("Superclasse: " + super.getName());
        if (clazz != Object.class) {
            imprimirHierarquia(super);
        }
    }
}
```



base
util
Map<K,V>

ct
AbstractMap<K,V>
util.TreeMap<K,V>

rs:

keys maintained by this map

V - the type of mapped values

All Implemented Interfaces:

Serializable, Cloneable, Map<K,V>, NavigableMap<K,V>, SortedMap<K,V>

```
public class TreeMap<K,V>
    extends AbstractMap<K,V>
    implements NavigableMap<K,V>, Cloneable, Serializable
```

Exemplo: hierarquia de classes

```
private static void imprimirHierarquia(Class<?> c, int nivel)
{
    List<Class<?>> lista = getSuperclasseEInterfaces(c);
    String recuo = "";
    for (int i=0; i<nivel; i++) { recuo += "    "; }
    for (Class<?> clazz : lista) {
        System.out.print(recuo+"|-> "+clazz.getName());
        if (!clazz.isInterface()) { // se não é interface...
            System.out.print(" (super)"); //... é a classe mãe
        }
        System.out.println(" ");
        if (clazz != Object.class) {
            imprimirHierarquia(clazz, nivel+1);
        }
    }
}
```

Montamos uma lista de classes ancestrais e interfaces Implementadas pela classe do parâmetro (função definida em breve)

Exemplo: hierarquia de classes

```
private static void imprimirHierarquia(Class<?> c, int nivel)
{
    List<Class<?>> lista = getSuperclasseEInterfaces(c);
    String recuo = "";
    for (int i=0; i<nivel; i++) { recuo += "  "; }
    for (Class<?> clazz : lista) {
        System.out.print(recuo+"| -> "+clazz.getName());
        if (!clazz.isInterface()) { // se não é interface...
            System.out.print(" (super)"); //... é a classe mãe
        }
        System.out.println(" ");
        if (clazz != Object.class) {
            imprimirHierarquia(clazz, nivel+1);
        }
    }
}
```

Uma variável com espaços
controla a indentação
conforme o nível

Exemplo: hierarquia de classes

```
private static void imprimirHierarquia(Class<?> c, int nivel)
{
    List<Class<?>> lista = getSuperclasseEInterfaces(c);
    String recuo = "";
    for (int i=0; i<nivel; i++) { recuo += "    "; }
    for (Class<?> clazz : lista) {
        System.out.print(recuo+"|-> "+clazz.getName());
        if (!clazz.isInterface()) { // se não é interface...
            System.out.print(" (super)"); //... é a classe mãe
        }
        System.out.println(" ");
        if (clazz != Object.class) {
            imprimirHierarquia(clazz, nivel+1);
        }
    }
}
```

Percorrendo a lista, são exibidos os espaços e o nome da classe ou interface

Exemplo: hierarquia de classes

```
private static void imprimirHierarquia(Class<?> c, int nivel)
{
    List<Class<?>> lista = getSuperclasseEInterfaces(c);
    String recuo = "";
    for (int i=0; i<nivel; i++) { recuo += "  "; }
    for (Class<?> clazz : lista) {
        System.out.print(recuo+"| -> "+clazz.getName());
        if (!clazz.isInterface()) { // se não é interface...
            System.out.print(" (super)"); //... é a classe mãe
        }
        System.out.println(" ");
        if (clazz != Object.class) {
            imprimirHierarquia(clazz, nivel+1);
        }
    }
}
```

Se não for interface, é
uma classe ancestral
(pai)

Exemplo: hierarquia de classes

```
private static void imprimirHierarquia(Class<?> c, int nivel)
{
    List<Class<?>> lista = getSuperclasseEInterfaces(c);
    String recuo = "";
    for (int i=0; i<nivel; i++) { recuo += "    "; }
    for (Class<?> clazz : lista) {
        System.out.print(recuo+"|-> "+clazz.getName());
        if (!clazz.isInterface()) { // se não é interface...
            System.out.print(" (super)"); //... é a classe mãe
        }
        System.out.println(" ");
        if (clazz != Object.class) {
            imprimirHierarquia(clazz, nivel+1);
        }
    }
}
```

Se a classe atual ainda não é a classe raiz ("pai de todas"), exibe a hierarquia superior

Criando instância

Exemplo: criando instância

```
public static void main(String args[]) {  
    // classe local  
    class Pessoa {  
        String nome;  
        String telefone;  
        @Override  
        public String toString() {  
            return nome + ", " + telefone;  
        }  
    }  
  
    // instanciando normalmente  
    Pessoa joao = new Pessoa();  
    joao.nome = "Joao da Silva";  
    joao.telefone = "12341234";  
    System.out.println(joao);  
}
```

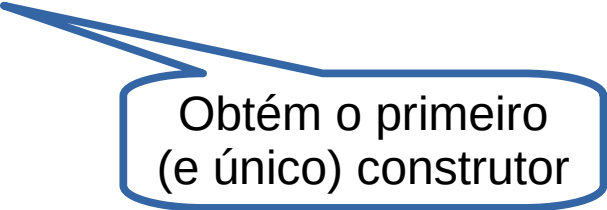
Exemplo: criando instância

```
Class<?> cla = Pessoa.class;  
try {  
    Constructor c = cla.getDeclaredConstructors()[0];  
    try {  
        Pessoa maria = (Pessoa) c.newInstance();  
        maria.nome = "Maria Oliveira";  
        maria.telefone = "91929394";  
        System.out.println(maria);  
    } catch ...  
} catch ...
```

Obtém referência
da classe

Exemplo: criando instância

```
Class<?> cla = Pessoa.class;
try {
    Constructor c = cla.getDeclaredConstructors()[0];
    try {
        Pessoa maria = (Pessoa) c.newInstance();
        maria.nome = "Maria Oliveira";
        maria.telefone = "91929394";
        System.out.println(maria);
    } catch ...
} catch ...
```



Obtém o primeiro
(e único) construtor

Exemplo: criando instância

```
Class<?> cla = Pessoa.class;
try {
    Constructor c = cla.getDeclaredConstructors()[0];
    try {
        Pessoa maria = (Pessoa) c.newInstance();
        maria.nome = "Maria Oliveira";
        maria.telefone = "91929394";
        System.out.println(maria);
    } catch ...
} catch ...
```



Cria a instância!

```
Joao da Silva, 12341234
Maria Oliveira, 91929394
```


Métodos úteis

Métodos úteis

`getFields()`  Atributos **públicos** da classe

Métodos úteis

`getFields()`  Atributos **públicos** da classe

`getDeclaredFields()`  **Todos** os atributos da classe

Métodos úteis

`getFields()` → Atributos **públicos** da classe

`getDeclaredFields()` → **Todos** os atributos da classe



Encontra atributos privados

Métodos úteis

```
getConstructors()
```

Construtores **públicos** na classe

Métodos úteis

```
getConstructors()
```

Construtores **públicos** na classe

```
getDeclaredConstructors()
```

Todos os construtores da classe

Métodos úteis

`getConstructors()`

Construtores **públicos** na classe

`getDeclaredConstructors()`

Todos os construtores da classe



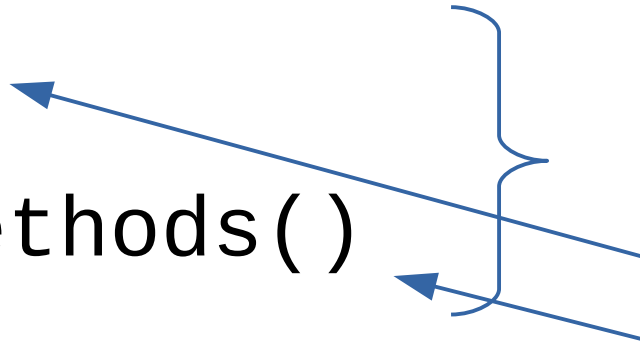
Inclusive o construtor “implícito” (padrão)

Métodos úteis

`getMethods()`

`getDeclaredMethods()`

Similar ao
retorno de
atributos:
públicos e
todos



Métodos úteis

`getParameterTypes()`

Retorna vetor com tipos de parâmetros que um método possui.

Métodos úteis

`getParameterTypes()`

Retorna vetor com tipos de parâmetros que um método possui.

`invoke()`

Executa um método!

Exemplo de execução de método

```
public static void main(String args[]) {  
    // classe local  
    class Pessoa {  
        String nome;  
        String telefone;  
  
        public void enviarMensagemWhats(String msg) {  
            System.out.println(nome+  
                " enviou via whats (" + telefone + "): " + msg);  
        }  
        @Override  
        public String toString() {  
            return nome + ", " + telefone;  
        }  
    }  
}
```

Exemplo de execução de método

```
Pessoa joao = new Pessoa();  
joao.nome = "Joao da Silva";  
joao.telefone = "12341234";  
System.out.println(joao);  
joao.enviarMensagemWhats("bom dia");
```

```
Method m;  
try {  
    m = acharMetodoPeloNome(Pessoa.class, "enviarMensagemWhats");  
    m.invoke(joao, "tudo bem com vocês?");  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

Método (descrito em breve) que busca um método de uma classe, a partir do nome do método

Exemplo de execução de método

```
Pessoa joao = new Pessoa();  
joao.nome = "Joao da Silva";  
joao.telefone = "12341234";  
System.out.println(joao);  
joao.enviarMensagemWhats("bom dia");
```

```
Method m;  
try {  
    m = acharMetodoPeloNome(Pessoa.class, "enviarMensagemWhats");  
    m.invoke(joao, "tudo bem com vocês?");  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

Execução do método, informando sobre
Qual objeto o método será executado e
Qual é o valor do parâmetro do método.

Exemplo de execução de método

```
Pessoa joao = new Pessoa();  
joao.nome = "Joao da Silva";  
joao.telefone = "12341234";  
System.out.println(joao);  
joao.enviarMensagemWhats("bom dia");
```

```
Method m;  
try {  
    m = acharMetodoPeloNome(Pessoa.class, "enviarMensagemWhats");  
    m.invoke(joao, "tudo bem com vocês?");  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

A execução poderia ser mais genérica,
buscando-se informações sobre quantos
parâmetros o método tem
(neste caso, apenas 1)

Exemplo de execução de método

```
Pessoa joao = new Pessoa();  
joao.nome = "Joao da Silva";  
joao.telefone = "12341234";  
System.out.println(joao);  
joao.enviarMensagemWhats("bom dia");
```

```
Method m;
```

```
try {  
    m = acharMetodoPeloNome(Pessoa.class, "enviarMensagemWhats");  
    m.invoke(joao, "tudo bem com vocês?");  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

```
public static Method acharMetodoPeloNome(  
    Class<?> c, String nome) throws Exception {  
    for (Method m : c.getMethods()) {  
        if (m.getName().equals(nome)) {  
            return m;  
        }  
    }  
    throw new Exception("Método " + nome + " não encontrado");  
}
```

Capítulo 3: metadados e anotações

Metadados e anotações

Metadados são dados que se referem aos próprios dados

Formas de informar metadados

Definição programática

Listagem 3.4 - Exemplo de configuração programática de metadados de persistência no framework Mentawai:

```
@Override
public void loadBeans() {
    bean(Usuario.class, "Usuarios")
        .pk("id", DBTypes.AUTOINCREMENT)
        .field("login", DBTypes.STRING)
        .field("senha", DBTypes.STRING)
        .field("dataNascimento", "nascimento", DBTypes.DATE);
}
```

Formas de informar metadados

Definição programática

Fontes externas

```
<hibernate-mapping>
  <class name="com.casadocodigo.Usuario" table="USUARIO">
    <id column="USER_ID" name="id" type="java.lang.Long">
      <generator class="org.hibernate.id.TableHiLoGenerator">
        <param name="table">idgen</param>
        <param name="column">NEXT</param>
      </generator>
    </id>
    <property column="LOGIN" name="login" type="java.lang.String"/>
    <property column="SENHA" name="senha" type="java.lang.String"/>
    <property column="NASCIMENTO" name="dataNascimento"
      type="java.util.Date"/>
  </class>
</hibernate-mapping>
```

Formas de informar metadados

Definição progr

Fontes externas

Anotações

Listagem 3.7 - Mapeamento objeto-relacional com JPA utilizando anotações:

```
@Entity  
@Table(name="Usuarios")  
public class Usuario {
```

```
@Id  
@GeneratedValue(strategy = GenerationType.IDENTITY)  
private int id;
```

```
private String login;  
private String senha;
```

```
@Column(name="nascimento")  
private Date dataNascimento;
```


Anotações

Elas não fazem nada!

Anotações

Elas não fazem nada!



Agregam informações às classes

Anotações

Elas não fazem nada!



Uma anotação é declarada com a expressão **@interface**

Agregam informações às classes



Um programa precisa usar essas informações

Tipo de anotações

SOURCE	Disponível no código fonte (tempo de compilação)
--------	---

Tipo de anotações

SOURCE	Disponível no código fonte (tempo de compilação)
CLASS (padrão)	Disponível até o momento do carregamento da classe

Tipo de anotações

Proc. of the 20th Conf. of Electrical Engineering, Bangkok, 1997

Post processing optimization of byte-code instructions by extension of its virtual machine.

SOURCE

Prabhas Chongstitvatana

Department of Computer Engineering, Chulalongkorn University,
Phaya Thai Road, Bangkok 10330, Thailand.

Phone (66-2) 218-3721, Fax (66-2) 215-3554, E-mail : fengpjs@chulkn.car.chula.ac.th

CLASS

(padrão)

Disponível até o momento do
carregamento da classe

Tipo de anotações

*O tipo da anotação é informado com anotação **@Retention***

SOURCE	Disponível no código fonte (tempo de compilação)
CLASS (padrão)	Disponível até o momento do carregamento da classe
RUNTIME	Disponível em tempo de execução, para reflexão :-)

Tipo de elementos anotados

TYPE

Classes, interfaces, enumerações...

PACKAGE

CONSTRUCTOR

FIELD

METHOD

PARAMETER

Parâmetros de métodos e construtores

LOCAL_VARIABLE

ANNOTATION_TYPE

Tipo de elementos anotados

TYPE

Classes, interfaces, enumerações...

PACKAGE

CONSTRUCTOR

FIELD

*O tipo da anotação é informado
com anotação @Target*

METHOD

PARAMETER

Parâmetros de métodos e construtores

LOCAL_VARIABLE

ANNOTATION_TYPE

Recuperação das anotações

`isAnnotationPresent(c classe)`

Existe anotação no elemento?

Recuperação das anotações

`isAnnotationPresent(classe)`

Existe anotação no elemento?

`getAnnotation(classe)`

Retorna uma anotação de uma class

Exemplos de anotações: definição

```
//anotação básica
public @interface Metadado {
}

//anotação disponível em tempo de execução
@Retention(RetentionPolicy.RUNTIME)
public @interface Metadado2 {
}

//anotação para método e atributo
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD, ElementType.FIELD})
public @interface Metadado3 {
}

//anotação com valor (parâmetro)
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.FIELD})
public @interface Metadado4 {
    int valor();
}
```

Exemplos de anotações: definição

//anotação básica

```
public @interface Metadado {  
}
```

//anotação disponível em tempo de execução

```
@Retention(RetentionPolicy.RUNTIME)
```

```
public @interface Metadado2 {  
}
```

//anotação para método e atributo

```
@Retention(RetentionPolicy.RUNTIME)
```

```
@Target({ElementType.METHOD, ElementType.FIELD})
```

```
public @interface Metadado3 {  
}
```

//anotação com valor (parâmetro)

```
@Retention(RetentionPolicy.RUNTIME)
```

```
@Target({ElementType.FIELD})
```

```
public @interface Metadado4 {  
    int valor();  
}
```

@Metadado

```
String atributoA;
```

@Metadado2

```
String atributoB;
```

@Metadado3

```
public String metodoA() {  
    return "ok";  
}
```

@Metadado4(valor = 10)

```
String atributoC;
```

Exemplos de anotações: uso

```
System.out.println("Exemplos de anotações");
Class<?> c = ExemplosAnotacoes.class;
try {
    Field f = c.getDeclaredField("atributoC");
    if (f.isAnnotationPresent(Metadado4.class)) {
        Metadado4 m4 = f.getAnnotation(Metadado4.class);
        System.out.println("Valor no metadado4: " + m4.valor());
    }
} catch (NoSuchFieldException | SecurityException e) {
    e.printStackTrace();
}
```

```
@Metadado4(valor = 10)
String atributoC;
```

Exemplos de anotações: uso

```
System.out.println("Exemplos de anotações");
Class<?> c = ExemplosAnotacoes.class;
try {
    Field f = c.getDeclaredField("atributoC");
    if (f.isAnnotationPresent(Metadado4.class)) {
        Metadado4 m4 = f.getAnnotation(Metadado4.class);
        System.out.println("Valor no metadado4: " + m4.valor());
    }
} catch (NoSuchFieldException | SecurityException e) {
    e.printStackTrace();
}
```

Obter a classe na qual
o atributo será buscado

```
@Metadado4(valor = 10)
String atributoC;
```

Exemplos de anotações: uso

```
System.out.println("Exemplos de anotações");
Class<?> c = ExemplosAnotacoes.class;
try {
    Field f = c.getDeclaredField("atributoC");
    if (f.isAnnotationPresent(Metadado4.class)) {
        Metadado4 m4 = f.getAnnotation(Metadado4.class);
        System.out.println("Valor no metadado4: " + m4.valor());
    }
} catch (NoSuchFieldException | SecurityException e) {
    e.printStackTrace();
}
```

Busca pelo atributo.
Usamos declared
Pois o atributo não
está público.

```
@Metadado4(valor = 10)
String atributoC;
```


Exemplos de anotações: uso

```
System.out.println("Exemplos de anotações");
Class<?> c = ExemplosAnotacoes.class;
try {
    Field f = c.getDeclaredField("atributoC");
    if (f.isAnnotationPresent(Metadado4.class)) {
        Metadado4 m4 = f.getAnnotation(Metadado4.class);
        System.out.println("Valor no metadado4: " + m4.valor());
    }
} catch (NoSuchFieldException | SecurityException e) {
    e.printStackTrace();
}
```

Verificamos se a anotação Metadado4 está presente no atributo encontrado

```
@Metadado4(valor = 10)
String atributoC;
```

Exemplos de anotações: uso

```
System.out.println("Exemplos de anotações");
Class<?> c = ExemplosAnotacoes.class;
try {
    Field f = c.getDeclaredField("atributoC");
    if (f.isAnnotationPresent(Metadado4.class)) {
        Metadado4 m4 = f.getAnnotation(Metadado4.class);
        System.out.println("Valor no metadado4: " + m4.valor());
    }
} catch (NoSuchFieldException | SecurityException e) {
    e.printStackTrace();
}
```

Obtemos acesso à
anotação

```
@Metadado4(valor = 10)
String atributoC;
```

Exemplos de anotações
Valor no metadado4: 10

Exemplos de anotações: uso

```
System.out.println("Exemplos de anotações");
Class<?> c = ExemplosAnotacoes.class;
try {
    Field f = c.getDeclaredField("atributoC");
    if (f.isAnnotationPresent(Metadado4.class)) {
        Metadado4 m4 = f.getAnnotation(Metadado4.class);
        System.out.println("Valor no metadado4: " + m4.valor());
    }
} catch (NoSuchFieldException | SecurityException e) {
    e.printStackTrace();
}
```

Por fim, o valor no parâmetro da anotação é exibido

```
@Metadado4(valor = 10)
String atributoC;
```

Exemplos de anotações
Valor no metadado4: 10

Exemplos de anotações: uso

```
// percorrer atributos
for (Field f : c.getDeclaredFields()) {
    System.out.println("Anotações do: "+f.getName());
    // obter anotações do atributo
    Annotation[] ans = f.getAnnotations();
    for (Annotation an : ans) {
        // exibir nome da anotação
        System.out.println(an.annotationType().getName());
    }
}
```

Exemplos de anotações: uso

```
// percorrer atributos
for (Field f : c.getDeclaredFields()) {
    System.out.println("Anotações do: "+f.getName());
    // obter anotações do atributo
    Annotation[] ans = f.getAnnotations();
    for (Annotation an : ans) {
        // exibir nome da anotação
        System.out.println(an.annotationType().getName());
    }
}
```

Percorrer atributos
da classe
(todos - *declared*)

Exemplos de anotações: uso

```
// percorrer atributos
for (Field f : c.getDeclaredFields()) {
    System.out.println("Anotações do: "+f.getName());
    // obter anotações do atributo
    Annotation[] ans = f.getAnnotations();
    for (Annotation an : ans) {
        // exibir nome da anotação
        System.out.println(an.annotationType().getName());
    }
}
```



Exibir o nome do atributo

Exemplos de anotações: uso

```
// percorrer atributos
for (Field f : c.getDeclaredFields()) {
    System.out.println("Anotações do: "+f.getName());
    // obter anotações do atributo
    Annotation[] ans = f.getAnnotations();
    for (Annotation an : ans) {
        // exibir nome da anotação
        System.out.println(an.annotationType().getName());
    }
}
```

Obter anotações do atributo. Poderia-se usar *Declared* para retornar também anotações de classes “super” (não é o caso)

Exemplos de anotações: uso

```
// percorrer atributos
for (Field f : c.getDeclaredFields()) {
    System.out.println("Anotações do: "+f.getName());
    // obter anotações do atributo
    Annotation[] ans = f.getAnnotations();
    for (Annotation an : ans) {
        // exibir nome da anotação
        System.out.println(an.annotationType().getName());
    }
}
```

Para cada anotação,
obter e exibir o **tipo**

```
Anotações do: atributoA
Anotações do: atributoB
reflexao.Metadado2
Anotações do: atributoC
reflexao.ExemplosAnotacoes$Metadado4
```


Exemplos de anotações: uso

```
// percorrer atributos
for (Field f : c.getDeclaredFields()) {
    System.out.println("Anotações do: "+f.getName());
    // obter anotações do atributo
    Annotation[] ans = f.getAnnotations();
    for (Annotation an : ans) {
        // exibir nome da anotação
        System.out.println(an.annotationType().getName());
    }
}
```

```
Anotações do: atributoA
Anotações do: atributoB
reflexao.Metadado2
Anotações do: atributoC
reflexao.ExemplosAnotacoes$Metadado4
```

Não aparece pois ficou com retenção padrão (CLASS)

```
//anotação básica
public @interface Metadado {
}
```

Exemplos de anotações: uso

```
// percorrer atributos
for (Field f : c.getDeclaredFields()) {
    System.out.println("Anotações do: "+f.getName());
    // obter anotações do atributo
    Annotation[] ans = f.getAnnotations();
    for (Annotation an : ans) {
        // exibir nome da anotação
        System.out.println(an.annotationType().getName());
    }
}
```

Anotação exibida

```
Anotações do: atributoA
Anotações do: atributoB
reflexao.Metadado2
Anotações do: atributoC
reflexao.ExemplosAnotacoes$Metadado4
```

```
@Retention(RetentionPolicy.RUNTIME)
public @interface Metadado2 {
}
```

Exemplos de anotações: uso

```
// percorrer atributos
for (Field f : c.getDeclaredFields()) {
    System.out.println("Anotações do: "+f.getName());
    // obter anotações do atributo
    Annotation[] ans = f.getAnnotations();
    for (Annotation an : ans) {
        // exibir nome da anotação
        System.out.println(an.annotationType().getName());
    }
}
```

- ExemplosAnotacoes.java
- Metadado2.java
- Metadado3.java
- Metadado.java

```
Anotações do: atributoA
Anotações do: atributoB
reflexao.Metadado2
Anotações do: atributoC
reflexao.ExemplosAnotacoes$Metadado4
```

Anotação definida no
mesmo arquivo da
classe

```
@Metadado4(valor = 10)
String atributoC;
```

FIM



Códigos disponíveis em
<http://github.com/hvescovi/dw2ed>

fundamentos > java > reflexao

<https://gitlab.com/hvescovi/prog24/-/tree/main/oo/java/reflection/netbeans>

