



Tutorial Java - Tiger

As novidades da versão 5.0 do J2SE

Igor Montezano dos Santos
Rodrigo Ferreira Jardim





Contexto Histórico

- Lançado em outubro de 2004
- Crecimento da plataforma M\$not.YET
- Alterações visam melhorias principalmente para o código





Principais novidades

- Generics
- Enums
- For each
- Anotations
- Var args
- Covariância
- Static Import
- Auto boxing/unboxing
- String Formater





Que porcaria é esta!?!?

<E> ?





Generics

- Tipagem de classes ou métodos em “build time”
- Elimina “casts” massivos
- Promove estruturas de dados fortemente tipadas





Enums

- Tipos enumerados (C, C++)
- Pattern TypeSafe Enum
- Vieram pra sepultar os antigos
“public static final int INFELIZ = 0;”





Erro possível!!

```
Calendar data = Calendar.getInstance();  
data.add(Calendar.MONTH, 1);
```

// Oque me impede de fazer isso...

```
data.add(-849, 1);
```





ForEach

- Iteração direta
- Arrays primitivos e Iterables
- Continuamos não podendo alterar as coleções





Antes e Depois!

```
Collection<String> c = new ArrayList<String>(50);
```

```
c.add("teste1"); c.add("teste2");
```

```
// assim era antes...
```

```
for(Iterator i=c.iterator();i.hasNext();){
```

```
    System.out.println(i.next());
```

```
}
```

```
// ... e agora!!
```

```
for(String s : c){
```

```
    System.out.println(s);
```

```
}
```





Covariância

- Especialização do tipo de retorno de um método sobrescrito
- Não realizado para parâmetros





Agora sim não compila!

```
Integer numero =  
100 / new Integer(4);
```





Auto boxing/unboxing

- Melhora convivência Objetos-Primitivos
- O código fica mais claro
- O bytecode gerado é compatível com versões anteriores
- Agora tudo virou uma maravilha?



Nada é de graça

```
public class Performance {
    public static void main(String args[]) {
        long inicio = Calendar.getInstance().getTimeInMillis();
        int io = 1;
        for(Integer i = 1; i < 100000; i++){
            io *= i;
        }
        System.out.println(io);
        long fim = Calendar.getInstance().getTimeInMillis();
        System.out.println(String.format("Auto Boxing/Unboxing %dms", fim-inicio));
        inicio = Calendar.getInstance().getTimeInMillis();
        io = 1;
        for(int i = 1; i < 100000; i++){
            io *= i;
        }
        System.out.println(io);
        fim = Calendar.getInstance().getTimeInMillis();
        System.out.println(String.format("Primitivo %dms", fim-inicio));
    }
}
```





Auto boxing/unboxing

- O código utilizando auto boxing/unboxing é convertido para casts em tempo de compilação
- Casts e criação de objetos custa tempo de processamento
- Não utilizar este tipo de solução para processamento científico



Olha só a zona!

```
public class PrimeiroContato {  
    private Integer inteiro = 5;  
    private int inteiro1 = new Integer(5);  
    private Map<String,Integer> mapa = new HashMap<String,Integer>();  
  
    public PrimeiroContato() {  
        System.out.println(inteiro.equals(inteiro1));  
        mapa.put("um",1);  
        mapa.put("dois",new Integer(2));  
        mapa.put("inteiro",inteiro);  
        mapa.put("inteiro1",inteiro1);  
        for(Map.Entry<String,Integer> entry : mapa.entrySet()){  
            String nome = entry.getKey();  
            int valor = entry.getValue();  
            System.out.println(String.format("%10s=%5d",nome,valor));  
        }  
    }  
}
```





E isto o que é?

@ooque?





Annotations

- Adição de metadados a linguagem
- A nova feature mais útil e mais inútil ao mesmo tempo
- Facilidade para criar novas anotações
- Anotações criadas não fazem absolutamente nada.
- Override, Deprecated e SuppressWarnings já vem por default e são tratadas pelo compilador





Quem já esta usando?

- Hibernate
 - Configuração de mapeamento
 - Validações
- EJB 3
- JDO
- JSR 250 – Commons Annotations for the Java Platform





Do inútil a mágica!

- Criando uma anotação
- Definindo o tipo de retenção e o alvo
- Se runtime deve ser utilizada via algum framework
- Se classe, pode ser processada pelo APT
- Se fonte, só pode ser utilizada pelo compilador ou algum tipo de pré processador





A mágica em Runtime

- Frameworks como o Hibernate utilizam anotações para configurar o comportamento das classes, no caso do Hibernate, para configurar o mapeamento O/R.
- Para obter este tipo de informação se utilizam os novos metodos da API de reflection do JDK.
- Anotações runtime podem ser utilizadas para prover informações para engines de AOP (exemplo o controle de transações via anotações do spring)





A mágica pelo APT

- O APT pode ser utilizado para gerar código fonte baseado em anotações
- Verificar padronizações de código e configurar o ambiente
- Pode ser bastante útil para a geração de configurações





String Formater/Scanner

- Facilidades do C/C++ agora disponíveis em java: Formatter e Scanner
- Similares a sprintf e sscanf, e utilizam as mesmas strings de formatação com um toque de regular expressions
- Muito mais fácil para formatar strings, e principalmente para buscar dados em uma string ou arquivo
- Metodos helpers distribuidos por conveniencia pelo JDK





Ta, e ai?

```
public class PrimeiroContato {
    public static void main(String[] args) {
        String input = "1 peixe -2 peixe vermelho peixe azul peixe";
        Scanner s = new Scanner(input).useDelimiter("\\s*peixe\\s*");
        while (s.hasNextInt())
            System.out.printf("Inteiro lido %30d\n", s.nextInt());
        while (s.hasNext())
            System.out.printf("String lida %30s\n", s.next());
        s.close();
    }
}
```

----- Saída -----	
Inteiro lido	1
Inteiro lido	(2)
String lida	vermelho
String lida	azul





Formatos Suportados

- Baseados no printf do C, mas não 100% compatível
- Suporte nativo para Numeros, Strings, e tipos Date/Time
- Tipos personalizados via `java.util.Formatter`
- Documentação completa no javadoc das classes `Formatter` e `Scanner`





Var Args

- Similar aos recursos já disponíveis em C
- Traduzido em tempo de compilação para um array
- Intercambiável por arrays, mas com a sintaxe mais limpa



Confundindo o usuário



```
public class PrimeiroContato {
    private void meuMetodoTeste(Object... args) {
        for(Object o: args){
            System.out.printf("Objeto tipo: %50s valor:
            %s\n",o.getClass().getName(),o.toString());
        }
    }

    private void meuMetodoTeste(Integer... args) {
        for(int i:args){
            System.out.printf("Inteiro %10d\n",i);
        }
    }

    public static void main(String[] args) {
        PrimeiroContato pc = new PrimeiroContato();
        pc.meuMetodoTeste(1, 2, 5, "teste", 5.3, new Date(), pc);
        pc.meuMetodoTeste(1, 2, 5);
    }
}
```

