

## Fundamentos Java e Orientação a Objetos

Por  
**Thiago Faria**

### 4.3. Desafio: wrappers e boxing

4. Wrappers, boxing e arrays



Seu chefe, que também adora programar, pediu sua ajuda para analisar o código que ele desenvolveu usando classes wrapper do java.lang. O código parece funcionar, mas ele gostaria de uma "consultoria" para saber se está usando as melhores práticas de programação e se no futuro não poderia dar nenhum problema.

Veja o código de uma classe do seu chefe:

```
class Televisor {  
  
    Integer canal = 130;  
    Integer volume = 20;  
  
    void mudarCanal(Integer novoCanal) {  
        if (canal == novoCanal) {  
            System.out.println("Novo canal é também o canal atual.");  
        } else {  
            canal = novoCanal;  
            System.out.println("Canal alterado para " + canal);  
        }  
    }  
  
    void mudarVolume(Integer novoVolume) {  
        if (novoVolume == volume) {  
            System.out.println("Novo volume é também o volume atual.");  
        } else {  
            volume = new Integer(novoVolume.byteValue());  
            System.out.println("Volume alterado para " + volume);  
        }  
    }  
  
    // Estamos aproveitando a classe Televisor para fazer nosso teste  
    public static void main(String[] args) {  
        Televisor tv = new Televisor();  
  
        // Não deveria mudar o volume e canal  
        tv.mudarVolume(20);  
        tv.mudarCanal(130);  
  
        // Deveria mudar o volume e canal  
        tv.mudarVolume(300);  
        tv.mudarCanal(10);  
    }  
}
```

O que você aprendeu neste curso que poderia usar para aproveitar essa oportunidade de mostrar para seu chefe que você realmente sabe Java?

Execute o código em seu computador e faça os ajustes que você achar necessário. Não deixe de comentar e interagir com os outros alunos que também estão estudando. :)

[Acesse o código-fonte desta aula](#)

#### Comentários sobre esta aula

**Paulo Mauricio Salles Rodrigue** - 08/05/2012 às 19:03

Mudei de == para equals e ficou mais certo. Não ficou totalmente certo, porque o volume deveria mudar para 300 e ficou 44. Mudei o volume para 200 e ficou -56. Como esse cálculo é feito?

Saida do programa, conforme o exercício:  
Novo volume:20. Novo volume é também o volume atual.  
Canal alterado para 130  
Volume alterado para 44 (usando o volume como 300)  
Canal alterado para 10

Saida do programa, conforme a alteração (usando o equals):  
Novo volume:20. Novo volume é também o volume atual.  
Novo canal:130. Novo canal é também o canal atual.  
Volume alterado para -56 (usando o volume como 200)  
Canal alterado para 10

**Normandes Júnior** INSTRUCTOR - 08/05/2012 às 23:44

Paulo, não entendi sua dúvida.  
Não era para ficar 300 em momento nenhum, e nem 44. O volume não é aumentado e nem diminuído, simplesmente é configurado.  
Como você entendeu o exercício, me explique melhor sua dúvida.

**Paulo Mauricio Salles Rodrigue** - 09/05/2012 às 12:16



Quando eu coloquei o novo volume como 300, o system.out.println colocou o volume alterado para 44. Quando eu coloquei o novo volume como 200, o system.out.println colocou o volume alterado para -56. Eu não entendi esse cálculo. Veja a cópia da saída do sistema, na mensagem acima. Abraço.



**Normandes Júnior** INSTRUTOR - 09/05/2012 às 14:38

Então, não existe cálculo. Se você trocou o "==" por ".equals" você simplesmente altera o valor do volume. Lembre-se que você também precisa alterar o código abaixo:

```
volume = new Integer(novoVolume.byteValue());  
  
por  
volume = novoVolume;
```

Pois o novoVolume.byteValue() irá truncar o valor do novo volume. Lembre-se que o byte vai de -128 a 127.



**Diogo Álvaro Bezerra** - 17/03/2012 às 17:09

Bom, primeiramente eu dividi o código em dois arquivos. No primeiro deixei só a classe Televisor e criei outro arquivo chamado "Principal.java".

Na classe Televisor fiz as mesmas alterações que o colega Manuel Monteiro fez:

1) Troquei :

```
if (canal == novoCanal) {  
  
por  
if (canal.equals(novoCanal)) {
```

Fiz o mesmo com o "if" que comparar o volume.

Também alterei essa parte:

```
volume = new Integer(novoVolume.byteValue());  
  
por  
volume = novoVolume;
```

Converter o valor da variável 'novoVolume' para o tipo 'byte' fez ele perder precisão e por isso o resultado saia errado.



**Normandes Júnior** INSTRUTOR - 17/03/2012 às 18:23

É isso aí Diogo, muito bom.



**Manuel Monteiro** - 13/02/2012 às 09:54

```
public class Televisor  
{  
    Integer canal =130;  
    Integer volume = 20;  
  
    void mudarCanal (Integer novoCanal)  
    {  
        // fiz a alteração no metodo de compara para o metodo equals eem ves de ( ==)  
        if (canal.equals(novoCanal))  
        {  
            System.out.println("Novo canal é tambem o canal actual");  
        }  
        else  
        {  
            canal = novoCanal;  
            System.out.println("Canal alterado para " + canal);  
        }  
    }  
  
    void mudarVolume (Integer novoVolume)  
    {  
        // fiz a alteração no metodo de compara para o metodo equals eem ves de ( ==)  
        if (novoVolume.equals(volume))  
        {  
            System.out.println("Novo volume é também o volume atual.");  
        }  
        else  
        {  
  
            // Alterar a linha volume = new Integer(novoVolume.byteValue()); para a lista abaixo  
  
            volume = novoVolume;  
            System.out.println("Volume alterado para " + volume);  
        }  
    }  
  
    public static void main(String[] args)  
    {  
        Televisor tv = new Televisor();  
  
        // Não deveria mudar o volume e canal  
        tv.mudarVolume(20);  
        tv.mudarCanal(130);  
    }  
}
```

```
// Deveria mudar o volume e canal
tv.mudarVolume(300);
tv.mudarCanal(10);
```

```
}

}
```

és as correções necessaria para o exercicio acima e os devidos devido comentarios.

Compartilhe esta aula com seus amigos

[Twitter](#)[Facebook](#)

### 1. Introdução

- [1.1. Como aprender Java?](#) 5m 50s GRÁTIS
- [1.2. A história do Java](#) 2m 46s GRÁTIS
- [1.3. As plataformas Java e como elas evoluem](#) 10m 31s GRÁTIS
- [1.4. Máquina virtual Java](#) 8m 45s GRÁTIS
- [1.5. Baixando, instalando e configurando a JDK](#) 7m 59s GRÁTIS
- [1.6. Exercício: instalação da JDK](#) GRÁTIS

### 2. Fundamentos da linguagem

- [2.1. Codificando, compilando e executando o programa "oi mundo"](#) 13m 10s GRÁTIS
- [2.2. Exercício: codificando um primeiro programa](#) GRÁTIS
- [2.3. Comentários](#) 3m 3s GRÁTIS
- [2.4. Sequências de escape](#) 5m 14s GRÁTIS
- [2.5. Palavras reservadas](#) 3m 32s GRÁTIS
- [2.6. Convenções de código](#) 2m 28s GRÁTIS
- [2.7. Trabalhando com variáveis](#) 6m 18s GRÁTIS
- [2.8. Nomeando variáveis](#) 5m 42s GRÁTIS
- [2.9. Operadores aritméticos](#) 9m 36s GRÁTIS
- [2.10. Exercício: variáveis e operadores aritméticos](#) GRÁTIS
- [2.11. Tipos primitivos](#) 12m 0s GRÁTIS
- [2.12. Outros operadores de atribuição](#) 4m 43s GRÁTIS
- [2.13. Conversão de tipos primitivos](#) 12m 39s GRÁTIS
- [2.14. Promoção aritmética](#) 6m 25s GRÁTIS
- [2.15. Exercício: tipos primitivos e outros operadores de atribuição](#) GRÁTIS
- [2.16. Trabalhando com strings](#) 7m 5s GRÁTIS
- [2.17. Recebendo entrada de dados](#) 7m 41s GRÁTIS
- [2.18. Operadores de comparação e igualdade](#) 6m 40s GRÁTIS
- [2.19. Estruturas de controle if, else if e else](#) 12m 23s GRÁTIS
- [2.20. Exercício: Strings, entrada de dados, operadores de comparação e if else](#) GRÁTIS
- [2.21. Escopo de variáveis](#) 6m 3s GRÁTIS
- [2.22. Operadores lógicos](#) 15m 13s GRÁTIS
- [2.23. Exercício: operadores lógicos](#) GRÁTIS
- [2.24. Estrutura de controle switch](#) 7m 10s GRÁTIS
- [2.25. Operador ternário](#) 6m 49s GRÁTIS
- [2.26. Operadores de incremento e decremento](#) 8m 11s GRÁTIS
- [2.27. Estrutura de controle while](#) 5m 45s GRÁTIS
- [2.28. Estrutura de controle do-while](#) 3m 47s GRÁTIS
- [2.29. Estrutura de controle for](#) 4m 15s GRÁTIS
- [2.30. Cláusulas break e continue](#) 7m 2s GRÁTIS
- [2.31. Exercício: operador ternário, decremento e estruturas de repetição](#) GRÁTIS
- [2.32. Introdução e instalação do Eclipse IDE](#) 13m 40s GRÁTIS
- [2.33. Depurando códigos com o Eclipse](#) 8m 43s GRÁTIS
- [2.34. Exercício: instalando o Eclipse IDE](#) GRÁTIS

### 3. Orientação a Objetos - parte 1

- [3.1. O que é POO?](#) 2m 57s GRÁTIS
- [3.2. Classes e objetos](#) 5m 16s GRÁTIS
- [3.3. Criando uma classe com atributos](#) 2m 48s GRÁTIS
- [3.4. Instanciando objetos](#) 7m 59s GRÁTIS
- [3.5. Acessando atributos de objetos](#) 8m 32s GRÁTIS
- [3.6. Exercício: instanciando e acessando atributos do objeto](#) GRÁTIS
- [3.7. Composição de objetos](#) 9m 28s GRÁTIS
- [3.8. Valores padrão](#) 5m 59s GRÁTIS
- [3.9. Variáveis referenciam objetos](#) 9m 22s GRÁTIS
- [3.10. Criando, nomeando e chamando métodos](#) 8m 2s GRÁTIS
- [3.11. Métodos com retorno](#) 11m 13s GRÁTIS
- [3.12. Passando argumentos para métodos](#) 5m 25s GRÁTIS
- [3.13. Argumentos por valor ou referência](#) 7m 0s GRÁTIS
- [3.14. Exercício: composição de objetos e chamada de métodos](#) GRÁTIS

### 4. Wrappers, boxing e arrays

- [4.1. Wrappers do java.lang](#) 3m 31s GRÁTIS
- [4.2. Boxing](#) 6m 47s GRÁTIS
- [4.3. Desafio: wrappers e boxing](#) GRÁTIS
- [4.4. Trabalhando com arrays](#) 16m 37s GRÁTIS
- [4.5. Exercício: arrays](#) GRÁTIS

### 5. Orientação a Objetos - parte 2

5.1. Introdução à UML e diagrama de classes

7m 31sGRÁTIS

5.4. Construtores

11m 43sGRÁTIS

5.7. Desafio: objeto this, construtores e JavaBeans

GRÁTIS

5.10. Modificadores static e final

12m 40sGRÁTIS

5.13. Desafio: pacotes e enumerações

GRÁTIS

5.16. Sobreposição

7m 48sGRÁTIS

5.19. Exercício: sobrecarga

GRÁTIS

5.22. Desafio: polimorfismo e classes abstratas

GRÁTIS

5.2. Desafio: diagrama de classes

GRÁTIS

5.5. Encapsulamento e modificadores de acesso public e private

11m 7sGRÁTIS

5.8. Organizando os projetos em pacotes

11m51sGRÁTIS

5.11. Desafio: static e final

GRÁTIS

5.14. Herança e modificador protected

10m 42sGRÁTIS

5.17. Desafio: herança e sobreposição

GRÁTIS

5.20. Polimorfismo, casting de objetos e instanceof

18m 49sGRÁTIS

5.23. Interfaces

11m 49sGRÁTIS

5.3. O objeto this

8m 18sGRÁTIS

5.6. Criando JavaBeans

8m 40sGRÁTIS

5.9. Modificador de acesso default

6m 55sGRÁTIS

5.12. Enumerações

17m 26sGRÁTIS

5.15. Classe java.lang.Object

4m 13sGRÁTIS

5.18. Sobrecarga

7m 48sGRÁTIS

5.21. Classes abstratas

9m 49sGRÁTIS

5.24. Exercício: interfaces e polimorfismo

GRÁTIS

6. Tópicos avançados

6.1. Coleta de lixo

8m 40sGRÁTIS

6.4. Tratando e lançando exceções

29m 12sGRÁTIS

6.7. Trabalhando com datas

19m 28sGRÁTIS

6.10. Desafio: números

GRÁTIS

6.13. Arquivos JAR

6m 19sGRÁTIS

6.16. Desafio: javadoc

GRÁTIS

6.2. Classe java.lang.Math

16m 6sGRÁTIS

6.5. Desafio: exceções

GRÁTIS

6.8. Desafio: datas

GRÁTIS

6.11. Collections Framework

22m 25sGRÁTIS

6.14. Exercício: arquivos JAR

GRÁTIS

6.17. Próximos passos

4m 8sGRÁTIS

6.3. Desafio: classe java.lang.Math

GRÁTIS

6.6. Classes String, StringBuffer e StringBuilder

8m 26sGRÁTIS

6.9. Trabalhando com números

9m 12sGRÁTIS

6.12. Desafio: collections

GRÁTIS

6.15. Documentação javadoc

9m 55sGRÁTIS

6.18. Conclusão

2m 6sGRÁTIS

Cursos online

Depoimentos de alunos

Sobre nós

Cursos presenciais

Instrutores

Fale conosco

Apostilas gratuitas

Trabalhe conosco

AlgaWorks Softwares, Treinamentos e Serviços Ltda  
Av. Afonso Pena, 3538, Átrio Business Center  
CEP: 38400-710 - Uberlândia/MG - Brasil  
Tel. +55 (34) 8400-6931 - comercial@algaworks.com