Programação Orientada a Objetos

Prof. Paulo Henrique Pisani

http://professor.ufabc.edu.br/~paulo.pisani/

UFABC

agosto/2018

Tópicos

- clone()
- Cópia rasa vs cópia profunda
- Construtor de cópia

UFABC

clone()

UFABC

clone()

- Método da classe **Object**;
- Retorna uma **cópia** do objeto.

```
protected Object clone() throws CloneNotSupportedException
```

- Para que a chamada a esse método não lance exceção (CloneNotSupportedException), deve-se implementar a interface **Cloneable.**

UFABC

Vamos clonar uma prova!

- Utilizaremos o código da **Aula 5 - projProva** (com algumas adaptações);

Observe que a interface Cloneable deve ser implementada

```
package autocorrecao;

public class Prova implements Cloneable {

    ...

    @Override
    public Prova clone() throws CloneNotSupportedException {
        return (Prova)super.clone();
    }
}
```

Chama o clone() de Object, que faz uma cópia “rasa” do objeto.

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80");
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

prova.imprimirProva();
prova2.imprimirProva();
```

Copia a prova

O que será impresso?

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80");
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

prova.imprimirProva();
prova2.imprimirProva();
```

Copia a prova

Saída

```
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: Quanto eh 1 + 1?
- 80
- 2
- 700
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: Quanto eh 1 + 1?
- 80
- 2
- 700
```

UFABC

Vamos adicionar um método para modificar a cópia

```
package autocorrecao;

public class Prova implements Cloneable {

    private Pergunta[] perguntasDaProva = new Pergunta[2];

    ...

    public void apagarPerguntas() {
        for (int i = 0; i < perguntasDaProva.length; i++)
            perguntasDaProva[i] = null;
    }
}
```

Método para apagar todas as perguntas

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80");
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

prova2.apagarPerguntas();

prova.imprimirProva();
prova2.imprimirProva();
```

O que será impresso?

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80");
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

prova2.apagarPerguntas();

prova.imprimirProva();
prova2.imprimirProva();
```

Saída

```
--- PROVA ---
--- PROVA ---
```

UFABC

Ah!? Como assim? Apagou as duas provas!

Isso ocorreu porque fizemos uma **cópia rasa** do objeto prova. Os objetos que fazem parte da prova não foram copiados (apenas a referência foi copiada).

UFABC

Cópia rasa vs cópia profunda

UFABC

Cópia rasa vs cópia profunda

- **Cópia rasa (*shallow copy*):** copia apenas os valores de todos os atributos.
  - No caso de atributos de tipos primitivos (e.g. int, double), o **valor** é copiado;
  - No caso de atributos que são instâncias de classes (e.g. classe Pergunta), o **valor** do ponteiro é copiado (**mas o objeto referenciado é o mesmo!**);

- **Cópia profunda (*deep copy*):** faz uma cópia do objeto e de todos os objetos referenciados em atributos.

UFABC

Ok, vamos melhorar esse clone()

- Agora o vetor é clonado. Todo vetor possui clone() implementado (**mas ele faz cópia rasa do vetor também**).

```
@Override
public Prova clone() throws CloneNotSupportedException {
    Prova novaProva = (Prova)super.clone();
    novaProva.perguntasDaProva = this.perguntasDaProva.clone();
    return novaProva;
}
```

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80");
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

prova2.apagarPerguntas();

prova.imprimirProva();
prova2.imprimirProva();
```

O que será impresso?

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80");
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

prova2.apagarPerguntas();

prova.imprimirProva();
prova2.imprimirProva();
```

Legal, agora só a prova2 está em branco.

Saída

```
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: Quanto eh 1 + 1?
- 80
- 2
- 700
--- PROVA ---
```

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80");
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

//prova2.apagarPerguntas();
p2.setEnunciado("10 x 8?");
prova.imprimirProva();
prova2.imprimirProva();
```

O que será impresso?

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80")
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

//prova2.apagarPerguntas();
p2.setEnunciado("10 x 8?");
prova.imprimirProva();
prova2.imprimirProva();
```

Veja que o enunciado nas duas provas foi alterado!

Isso ocorreu porque a cópia do vetor de perguntas foi rasa.

Saída

```
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: 10 x 8?
- 80
- 2
- 700
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: 10 x 8?
- 80
- 2
- 700
```

UFABC

Tornando a cópia profunda...

- Para a cópia ser de fato profunda, é necessário que todos os objetos sejam copiados (e não apenas as referências).
- No caso, teríamos que copiar cada elemento do vetor de perguntas + implementar **clone()** nas classes Pergunta e Alternativa.

UFABC

Alternativa: clone()

```
package autocorrecao;

public class Alternativa implements Cloneable {

    private String textoAlternativa;

    ...

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

UFABC

Alternativa: clone()

Você não copiou o textoAlternativa! É um objeto String! Então só a referência foi copiada no super.cone()!

```
package autocorrecao;

public class Alternativa implements Cloneable {

    private String textoAlternativa;

    ...

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

UFABC

Alternativa: clone()

```
package autocorrecao;

public class Alternativa implements Cloneable {

     private String textoAlternativa;

     ...

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

Você não copiou o textoAlternativa! É um objeto String! Então só a referência foi copiada no super.cone()!

Sim, isso é verdade! Mas o objeto **String é imutável**, então não há necessidade de copiar um objeto que não pode ser alterado.

UFABC

Pergunta: clone()

```
package autocorrecao;

public class Pergunta implements Cloneable {

    private String enunciado;
    private Alternativa[] alternativas;

    ...

    @Override
    public Pergunta clone() throws CloneNotSupportedException {
        Pergunta novaPergunta = (Pergunta) super.clone();
        novaPergunta.alternativas = this.alternativas.clone();
        for (int i = 0; i < this.alternativas.length; i++)
            if (this.alternativas[i] != null)
                novaPergunta.alternativas[i] = this.alternativas[i].clone();
        return novaPergunta;
    }
}
```

UFABC

Prova: clone()

```
package autocorrecao;

public class Prova implements Cloneable {

    private Pergunta[] perguntasDaProva = new Pergunta[2];

    @Override
    public Prova clone() throws CloneNotSupportedException {
        //return (Prova)super.clone();
        Prova novaProva = (Prova)super.clone();
        novaProva.perguntasDaProva = this.perguntasDaProva.clone();
        for (int i = 0; i < this.perguntasDaProva.length; i++)
            if (this.perguntasDaProva[i] != null)
                novaProva.perguntasDaProva[i] = this.perguntasDaProva[i].clone();
        return novaProva;
    }
}
```

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80");
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

//prova2.apagarPerguntas();
p2.setEnunciado("10 x 8?");
prova.imprimirProva();
prova2.imprimirProva();
```

O que será impresso?

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80")
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

//prova2.apagarPerguntas();
p2.setEnunciado("10 x 8?");
prova.imprimirProva();
prova2.imprimirProva();
```

Veja que o enunciado apenas em uma prova foi afetado.

Saída

```
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: 10 x 8?
- 80
- 2
- 700
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: Quanto eh 1 + 1?
- 80
- 2
- 700
```

UFABC

E se quisermos proteger a prova, para não permitir que alterações externas afetem suas perguntas?

UFABC

Protegendo a prova contra alterações externas...

- Podemos copiar os objetos quando cada questão é adicionada;

```
public void adicionaPergunta(Pergunta novaPergunta)
    throws CloneNotSupportedException {
     boolean adicionouPergunta = false;
     for (int i = 0; i < perguntasDaProva.length
         && !adicionouPergunta; i++)
         if (perguntasDaProva[i] == null) {
             //perguntasDaProva[i] = novaPergunta;
             perguntasDaProva[i] = novaPergunta.clone();
             adicionouPergunta = true;
         }

     if (!adicionouPergunta)
         System.out.println("Nao adicionou pergunta!");
}
```

Copia cada pergunta

UFABC

```
Pergunta p1 = new Pergunta();
p1.setEnunciado("Quanto eh 2 + 2?");
Alternativa a1 = new Alternativa("80");
Alternativa a2 = new Alternativa("4");
p1.adicionaAlternativa(a1);
p1.adicionaAlternativa(a2);

Pergunta p2 = new Pergunta(3);
p2.setEnunciado("Quanto eh 1 + 1?");
Alternativa a3 = new Alternativa("80")
Alternativa a4 = new Alternativa("2");
p2.adicionaAlternativa(a3);
p2.adicionaAlternativa(a4);
p2.adicionaAlternativa("700");

Prova prova = new Prova();
prova.adicionaPergunta(p1);
prova.adicionaPergunta(p2);

Prova prova2 = prova.clone();

//prova2.apagarPerguntas();
p2.setEnunciado("10 x 8?");
prova.imprimirProva();
prova2.imprimirProva();
```

Agora as perguntas das duas provas estão protegidas

Saída

```
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: Quanto eh 1 + 1?
- 80
- 2
- 700
--- PROVA ---
Enunciado: Quanto eh 2 + 2?
- 80
- 4
Enunciado: Quanto eh 1 + 1?
- 80
- 2
- 700
```

UFABC

Simplificando o clone()

- Como agora cada pergunta já é copiada (**e não pode ser alterada**), podemos simplificar o clone() de Prova:

```
@Override
public Prova clone() throws CloneNotSupportedException {
    //return (Prova)super.clone();
    Prova novaProva = (Prova)super.clone();
    novaProva.perguntasDaProva = this.perguntasDaProva.clone();
    /*for (int i = 0; i < this.perguntasDaProva.length; i++)
        if (this.perguntasDaProva[i] != null)
            novaProva.perguntasDaProva[i] = this.perguntasDaProva[i].clone();
    */
    return novaProva;
}
```

Uma cópia rasa do vetor é suficiente agora!

UFABC

Qual cópia usar?

- Rasa:
  - **É mais rápida**: interessante quando os objetos referenciados nos atributos não são alterados;
- Profunda:
  - **Faz uma cópia completa (mais lenta)**: necessária quando os objetos referenciados nos atributos podem ser alterados.

UFABC

Construtor de cópia

UFABC

Construtor de cópia

- Outra forma de realizar cópia é com um contrutor de cópia;
- **É um construtor que recebe como argumento uma instância da própria classe**:
  - Esse construtor então copia todos os atributos;
  - Veja que o construtor de cópia pode ser implementado como cópia rasa ou profunda;

UFABC

Exemplo

- O construtor de cópia não necessita da implementação de Cloneable.

```
package autocorrecao;

public class Alternativa {
    private String textoAlternativa;

    public Alternativa(Alternativa a) {
        this.textoAlternativa = new String(a.textoAlternativa);
    }

    ...
}
```

Observe que a classe String já implementa um construtor de cópia

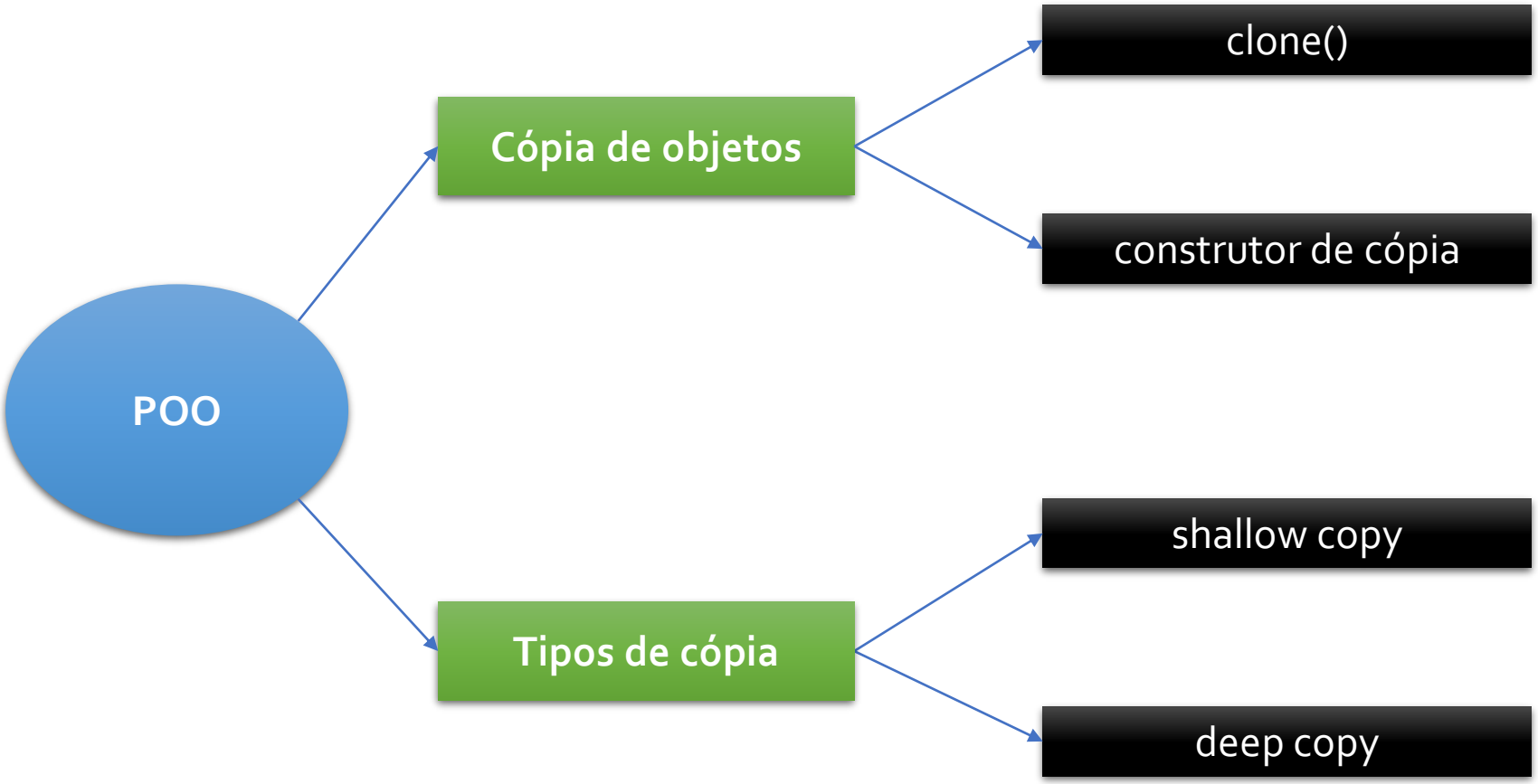
UFABC

clone() vs construtor de cópia

- O **método clone()** faz uma cópia rápida do objeto (não chama construtor para realizar a cópia):
  - Mas é um mecanismo de cópia automático (não temos controle de como ele é feito).
  - Envolve vários detalhes de sintaxe: interface Cloneable, exceção CloneNotSupportedException, cast do retorno de super.clone().
- O **construtor de cópia**, por outro lado, não possui os problemas mencionados acima:
  - Mas requer que seja feita a cópia de todos os atributos explicitamente.

UFABC

Resumo da aula



UFABC

Referências

- Documentação Java: https://docs.oracle.com/javase/8/docs/

UFABC

Referências (projeto pedagógico)

- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML: guia do usuário. Rio de Janeiro, RJ: Campus, 2005.
- GUEDES, G. T. A. UML 2: uma abordagem prática. São Paulo, SP: Novatec, 2009.
- DEITEL, H. M.; DEITEL, P. J. Java: como programar. 6a edição. Porto Alegre, RS: Bookman, 2005.
- BARNES, D. J.; KOLLING, M. Programação orientada a objetos com Java. 4ª edição. São Paulo, SP: Editora Pearson Prentice Hall, 2009.

UFABC

Referências (projeto pedagógico)

- FLANAGAN, D. Java: o guia essencial. 5ª edição. Porto Alegre, RS: Bookman, 2006.
- BRUEGGE, B.; DUTOIT, A. H. Object-oriented software engineering: using UML, patterns, and Java. 2ª edição. Upper Saddle River, NJ: Prentice Hall, 2003.
- LARMAN, C. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento interativo. 3ª edição. Porto Alegre, RS: Bookman, 2007.
- FOWLER, M. UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos. 3ª edição. Porto Alegre, RS: Bookman, 2005.

UFABC