Orientação a Objetos em Java

10ª aula

Prof. Douglas Oliveira
douglas.oliveira@prof.infnet.edu.br

Coleções - ArrayList

- Nos exemplos anteriores armazenamos na lista objetos já existentes na linguagem Java (Strings, caracteres, inteiros e reais).
- Entretanto, podemos armazenar na lista nossos próprios objetos:

```
public class Aluno {
  private int matricula;
  private String nome;
   public Aluno(int matricula, String nome) {
     this.matricula = matricula;
     this.nome = nome;
   }
   public String toString() {
     return String.format("Matricula: %d    Nome: %s\n", matricula,
nome);
   }
}
```

Coleções - ArrayList

- Exemplo 7: criando uma lista de alunos.

```
import java.util.ArrayList;
public class ExemploArrayList {
  public static void main(String[] args) {
      ArrayList alunos = new ArrayList();
      alunos.add(new Aluno(123, "Joao"));
      alunos.add(new Aluno(456, "Rosa"));
      alunos.add(new Aluno(789, "Ana"));
      alunos.add(new Aluno(246, "Lucas"));

      for (int i = 0; i < alunos.size(); i++)
          System.out.println(alunos.get(i));
  }
}
```

Ao imprimir um Aluno, será chamado o método **toString()** do objeto.

Coleções - ArrayList

- Importante: para usar os métodos:
  - ✔ `boolean remove(Object)`
  - ✔ `boolean contains(Object)`
  - ✔ `int indexOf(Object)`
  - ✔ `int lastIndexOf(Object)`

  as classes armazenadas no ArrayList tem que, obrigatoriamente, implementar o método

  `boolean equals(Object)`

- Caso esse método não seja implementado, os métodos acima não funcionarão corretamente. Isso acontece porque os métodos acima usam o método equals para procurar o objeto desejado.

Exercícios

- Exercício 45: altere o exercício dos alunos e curso e troque o vetor implementados nas classe Curso por uma lista. Não se esqueça de implementar o método equals() na classe Aluno.

Classes Wrapper

- Em Java, os tipos básicos int, long, byte, short, float, double, char e boolean não são objetos.
- Entretanto, a maioria das classes da biblioteca Java (como as classes que implementam as coleções, por exemplo) trabalha somente com objetos.
- Para resolver esse problema foram definidas classes *wrapper*, ou seja, classes que encapsulam os tipos básicos para que estes possam ser tratados como objetos.

Classe Wrapper	Tipo armazenado na classe
Integer	int
Long	long
Byte	byte
Short	short
Float	float
Double	double
Character	char
Boolean	boolean

Classes Wrapper

- Na maior parte das vezes, o Java converte o tipo básico em uma classe wrapper (*boxing*) ou uma classe wrapper em um tipo básico (*unboxing*) de forma automática:

```
int x = 10;
Integer a = 10;
Integer b = x + 1;
int y = a;
a = a + 1;
b++;
```

Classes Wrapper

- Nas classes *wrapper* estão definidos métodos para converter tipos básicos em classes (*boxing*) e classes em tipos básicos (*unboxing*).

Classe Wrapper	Tipo básico ☐ Classe (*boxing*)	Classe ☐ tipo básico (*unboxing*)
Integer	Integer n = Integer.valueOf(10)	int x = n.intValue()
Float	Float n = Float.valueOf(10.5f)	float x = n.floatValue()
Double	Double n = Double.valueOf(10.5)	double x = n.doubleValue()
Long	Long n = Long.valueOf(10)	long x = n.longValue()
Short	Short n = Short.valueOf(10)	short x = n.shortValue()
Byte	Byte n = Byte.valueOf(10)	byte x = n.byteValue()
Character	Character c = Character.valueOf('A')	char ch = c.charValue()
Boolean	Boolean b = Boolean.valueOf(true)	boolean bl = b.booleanValue()

Classes Wrapper

- Quando armazenamos inteiros, reais ou caracteres em uma lista, o Java faz automaticamente o *boxing*, ou seja, ele cria objetos usando as classes wrapper e insere esses objetos na lista.

```
import java.util.ArrayList;
public class ExemploArrayList {
  public static void main(String[] args) {
      ArrayList lista = new ArrayList();
      lista.add(10);
      lista.add(2465);
      lista.add(3.14159);
      lista.add('A');
      lista.add('C');
  }
}
```

Para armazenar esses valores são usadas as classes Integer, Double e Character.

Classes Wrapper

- Quando temos na lista objetos do tipo Integer, Double, Character, etc., temos que usar o *casting* para obter o valor armazenado dentro do objeto.

```
import java.util.ArrayList;
public class ExemploArrayList {
  public static void main(String[] args) {
      ArrayList lista = new ArrayList();
      lista.add(10);
      lista.add(2465);
      lista.add(3.14159);
      lista.add('A');
      lista.add('C');
      int n1    = (int) lista.get(0);
      int n2    = (int) lista.get(1);
      double n3 = (double) lista.get(2);
      char c1   = (char) lista.get(3);
      char c2   = (char) lista.get(4);
  }
}
```

Classes Wrapper

- As classes *wrapper* possuem, ainda, métodos estáticos úteis para converter strings em tipos específicos.
- Caso a conversão não seja possível, será disparada uma exceção.

Classe Wrapper	Conversão de String
Integer	int Integer.parseInt(String s)
Float	float Float.parseFloat(String s)
Double	double Double.parseDouble(String s)
Long	long Long.parseLong(String s)
Short	short Short.parseShort(String s)
Byte	byte Byte.parseByte(String s)

Coleções - Iterator

- Na interface Collection também é definido o método:
  - Iterator iterator(): retorna um objeto Iterator que permite percorrer os objeto da coleção de forma sequencial.

- A classe Iterator implementa os seguintes métodos:
  - ✔ boolean hasNext(): informa se existe um próximo elemento na coleção.
  - ✔ Object next(): retorna o próximo elemento da coleção.

- Como é possível perceber, o Iterator só permite varrer a coleção em uma única direção: do início para o fim.

Coleções - ListIterator

- Na interface List também é definido o método:
  - ListIterator listIterator(): retorna um objeto ListIterator que permite percorrer os objeto da lista da frente para trás ou vice-versa.
- A classe ListIterator possui todos os métodos da Iterator, mais:
  - boolean hasPrevious(): informa se existe um elemento anterior na lista.
  - Object previous(): retorna o elemento anterior da lista.
- Como é possível perceber, o ListIterator permite varrer a lista do início para o fim ou vice-versa.

Coleções - ArrayList

- Exemplo 8: usando um ListIterator.

```
import java.util.ArrayList;
public class ExemploArrayList {
  public static void main(String[] args) {
      ArrayList lista = new ArrayList();
      lista.add("Dinardo");
      lista.add("Rosa");
      lista.add(10);
      lista.add(2465);
      lista.add(3.14159);
      lista.add('A');
      ListIterator it = lista.listIterator();
      while (it.hasNext()) {
          Object obj = it.next();
          System.out.println(obj);
      }
  }
}
```