# Programação Orientada a Objetos

## Herança

Cristiano Lehrer, M.Sc.



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

ybadoo

# Introdução (1/4)

- **Herança** é um mecanismo que permite basear uma nova classe na definição de uma classe previamente existente.
  - Usando herança, a nova classe herda todos os atributos e comportamentos presentes na classe previamente existente.
  - Quando uma classe herda de outra, todos os métodos e atributos que aparecem na interface da classe previamente existente aparecerão automaticamente na interface da nova classe.



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

ybadoo

# Introdução (2/4)

```
public class Employee {

  private String first_name;
  private String last_name;
  private double wage;

  public Employee(String first_name, String last_name, double wage) {
    this.first_name = first_name;
    this.last_name = last_name;
    this.wage = wage;
  }

  public double getWage() {
    return this.wage;
  }

  public String getFirstname() {
    return this.first_name;
  }

  public String getLastname() {
    return this.last_name;
  }
}
```



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

ybadoo

# Introdução (3/4)

```
public class CommissionedEmployee extends Employee {

  private double commission;
  private int units;

  public CommissionedEmployee(String first_name, String last_name,
                              double wage, double commission) {
    super(first_name, last_name, wage);
    this.commission = commission;
  }

  public double calculatePay() {
    return getWage() + (this.commission * this.units);
  }

  public void addSales(int units) {
    this.units = this.units + units;
  }

  public void resetSales() {
    this.units = 0;
  }
}
```



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

# Introdução (4/4)

```
public static void main(String[] args) {

  CommissionedEmployee c;
  c = new CommissionedEmployee("Mr.", "Sales", 5.50, 1.00);
  c.addSales(5);

  System.out.println("First Name: " + c.getFirstname());
  System.out.println("Last Name: " + c.getLastname());
  System.out.println("Base Pay: $" + c.getWage());
  System.out.println("Total Pay: $" + c.calculatePay());
}
```

```
First Name: Mr.
Last Name: Sales
Base Pay: $5.5
Total Pay: $10.5
```



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

# É UM versus TEM UM

- **É um** descreve o relacionamento em que uma classe é considerada do mesmo tipo de outra.
- **Tem um** descreve o relacionamento em que uma classe contém uma instância de outra classe.
  - **Composição** significa que uma classe é implementada usando-se variáveis internas (chamadas de variáveis membro), que contêm instâncias de outras classes.
  - **Delegação** é o processo de um objeto passar uma mensagem para outro objeto, para atender algum pedido.



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

ybadoo

# Hierarquia (1/2)

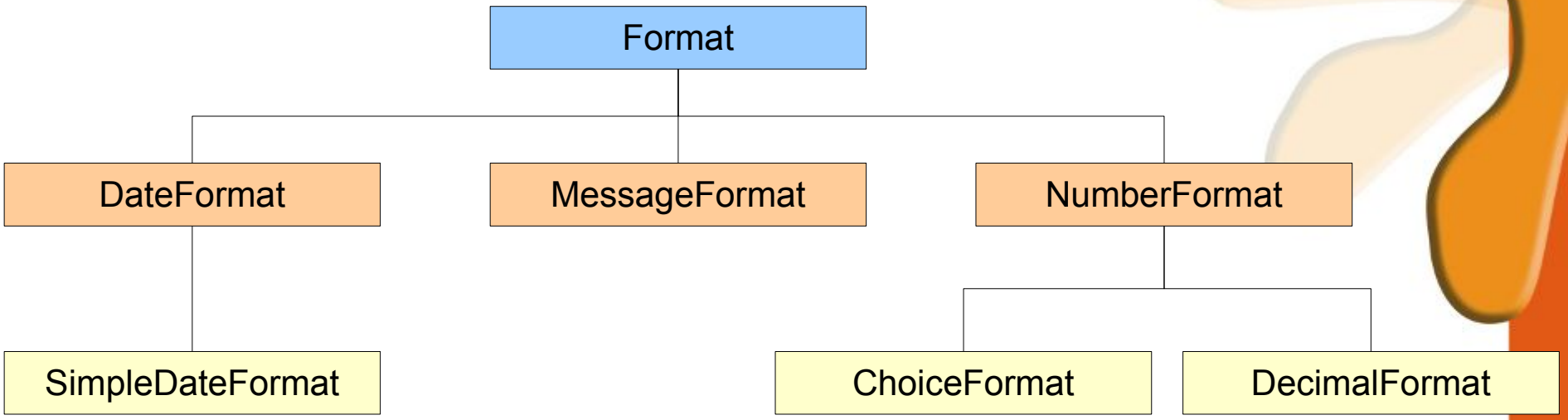
- Uma **hierarquia de herança** é um mapeamento do tipo árvore de relacionamentos que se formam entre classes como resultado da herança.
- A classe **filha** é a classe que está herdando, também conhecida como **subclasse**.
- A classe **progenitora** ou **mãe** é a classe da qual a filha herda diretamente, também conhecida como **superclasse**.
- Herança é um mecanismo que permite estabelecer relacionamentos 'é um' entre classes.
  - Esse relacionamento também permite que uma subclasse herde os atributos e comportamentos de sua superclasse.



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

ybadoo

# Hierarquia (2/2)






Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

ybadoo

# Mecânica da Herança (1/4)

- **Sobrepor** é o processo de uma filha pegar um método que aparece na progenitora e reescrevê-lo para mudar o comportamento do método.
  - A sobreposição de um método também é conhecida como redefinição de um método.



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

ybadoo

# Mecânica da Herança (2/4)

```
public class TwoDimensionalPoint {

  private double x, y;

  public TwoDimensionalPoint(double x, double y) {
    setXCoordinate(x);
    setYCoordinate(y);
  }

  public double getXCoordinate() { return this.x; }

  public double getYCoordinate() { return this.y; }

  public void setXCoordinate(double x) { this.x = x; }

  public void setYCoordinate(double y) { this.y = y; }

  public String toString() {
    return "I am a 2 dimensional point.\n" +
           "My x coordinate is: " + getXCoordinate() + "\n" +
           "My y coordinate is: " + getYCoordinate();
  }
}
```



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

# Mecânica da Herança (3/4)

```
public class ThreeDimensionalPoint extends TwoDimensionalPoint {

  private double z;

  public ThreeDimensionalPoint(double x, double y, double z) {
    super(x, y);
    setZCoordinate(z);
  }

  public double getZCoordinate() { return this.z; }

  public void setZCoordinate(double z) { this.z = z; }

  public String toString() {
    return "I am a 3 dimensional point.\n" +
           "My x coordinate is: " + getXCoordinate() + "\n" +
           "My y coordinate is: " + getYCoordinate() + "\n" +
           "My z coordinate is: " + getZCoordinate();
  }
}
```



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/

# Mecânica da Herança (4/4)

```
public static void main(String[] args) {

  TwoDimensionalPoint two = new TwoDimensionalPoint(1, 2);
  ThreeDimensionalPoint three = new ThreeDimensionalPoint(3, 4, 5);

  System.out.println(two.toString());
  System.out.println(three.toString());
}
```

```
    I am a 2 dimensional point.
    My x coordinate is: 1
    My y coordinate is: 2

    I am a 3 dimensional point.
    My x coordinate is: 3
    My y coordinate is: 4
    My z coordinate is: 5
```



Copyright © 2009/2016 • Ybadoo - Soluções em Software Livre • http://www.ybadoo.com.br/