







IMD0040  
 Linguagem de Programação 2



- Aula 08
- Aperfeiçoando estruturas com o uso da herança







## Nesta aula

- Aperfeiçoando estruturas com o uso da herança
  - Cap 8
- Conceitos
  - Herança, Subtipagem
  - Substituição, Variáveis polimórficas


## O exemplo do DoME

- Database of Multimedia Entertainment
- Armazena detalhes sobre CDs e DVDs
  - CD: título, artista, número de faixas, tempo de reprodução, flag got-it, comentário
  - DVD: título, nome do diretor, tempo de reprodução, flag got-it, comentário
- Permite pesquisar informações ou imprimir listas




## Objetos DoME

**:CD**

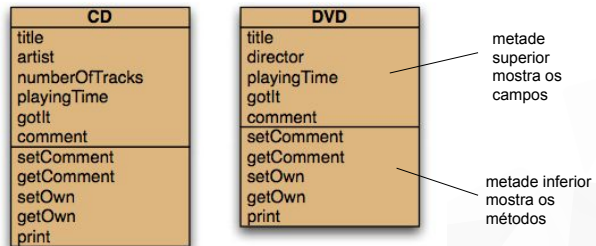
title	
artist	
#tracks	
playing time	
got it	
comment	

**:DVD**

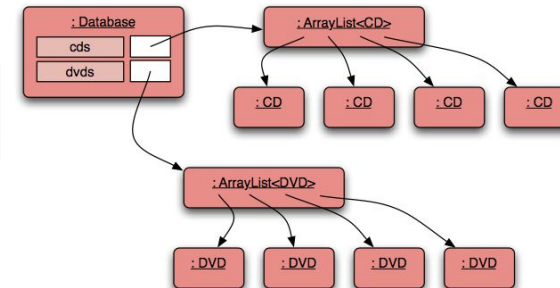
title	
director	
playing time	
got it	
comment	

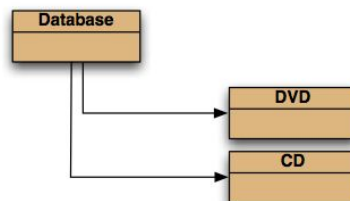

## Classes DoME



## Modelo de objetos DoME



## Diagrama de classes



## Código-fonte do CD

```

public class CD {
    private String title;
    private String artist;
    private String comment;

    public CD(String theTitle, String theArtist) {
        title = theTitle;
        artist = theArtist;
        comment = " ";
    }

    public void setComment(String newComment) { ... }
    public String getComment() { ... }
    public void print() { ... }
}
  
```

## Código-fonte do DVD

```
public class DVD {  
    private String title;  
    private String director;  
    private String comment;  
  
    public CD(String theTitle, String theDirector) {  
        title = theTitle;  
        director = theDirector;  
        comment = " ";  
    }  
    public void setComment(String newComment)  
    { ... }  
    public String getComment() { ... }  
    public void print() { ... }  
}
```

## Código-fonte de Database

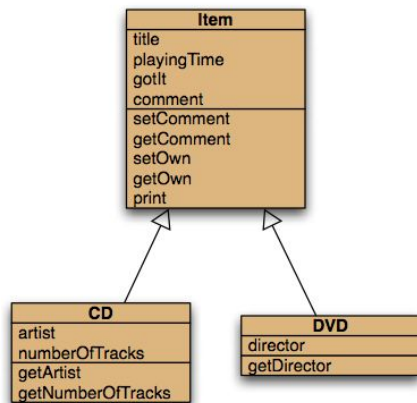
```
public class Database {  
    private ArrayList<CD> cds;  
    private ArrayList<DVD> dvds;  
  
    public void list() {  
        for (CD cd : cds) {  
            cd.print();  
            System.out.println(); //linha em branco entre itens  
        }  
  
        for (DVD dvd : dvds) {  
            dvd.print();  
            System.out.println(); //linha em branco entre itens  
        }  
    }  
}
```

## Crítica de DoME

- Duplicação de código
  - Classes CD e DVD muito semelhantes (grande parte é idêntica)
  - Torna manutenção difícil/mais trabalho
  - Introduz risco de bugs por meio de manutenção incorreta
- Duplicação de código também na classe Database

Utilizando herança

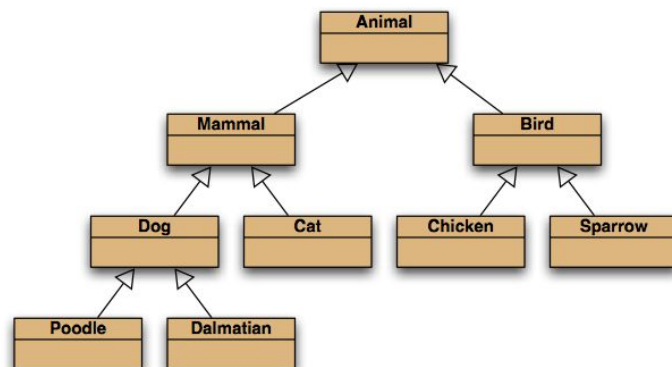
## Utilizando herança



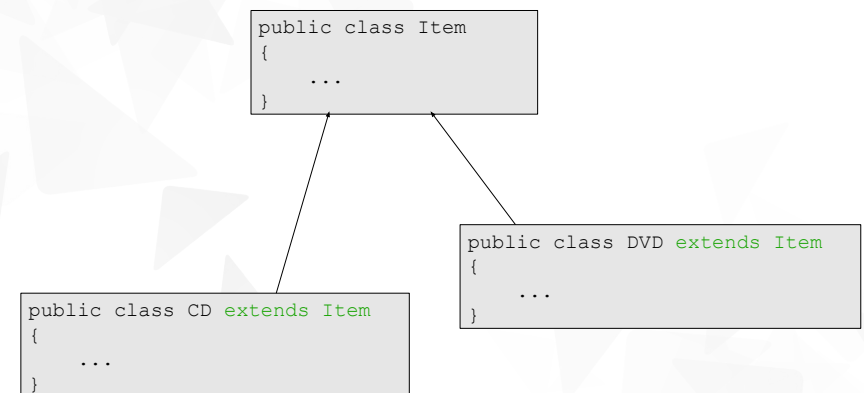
## Utilizando herança

- Define uma **superclasse**: item
- Define **subclasses** para DVD e CD
- A superclasse define atributos comuns
- As subclasses **herdam** os atributos da superclasse
- As subclasses adicionam atributos próprios

## Hierarquias de herança



## Herança em Java



## Superclasse

```
public class Item {
    private String title;
    private int playingTime;
    private boolean gotIt;
    private String comment;

    // construtores e métodos omitidos
}
```

## Subclasses

```
public class CD extends Item
{
    private String artist;
    private int numberOfTracks;

    // construtores e métodos omitidos
}
```

```
public class DVD extends Item
{
    private String director;

    // construtores e métodos omitidos
}
```

## Herança e construtores

```
public class Item
{
    private String title;
    private int playingTime;
    private boolean gotIt;
    private String comment;

    public Item(String theTitle, int time)
    {
        title = theTitle;
        playingTime = time;
        gotIt = false;
        comment = " ";
    }

    //métodos omitidos
}
```

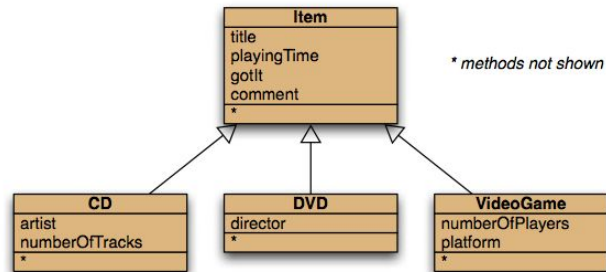
## Herança e construtores

```
public class CD extends Item
{
    private String artist;
    private int numberOfTracks;

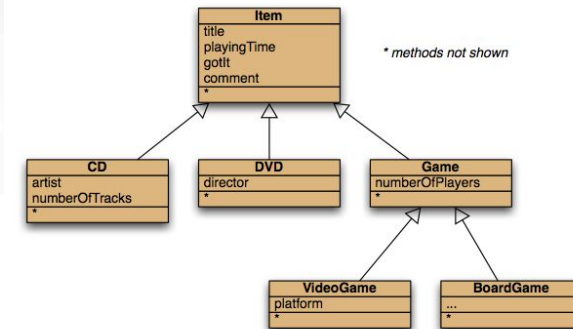
    public CD(String theTitle, String theArtist,
               int track, int time)
    {
        super(theTitle, time);
        artist = theArtist;
        numberOfTracks = tracks;
    }

    //métodos omitidos
}
```

## Adicionando mais tipos de item



## Hierarquias mais profundas



## Revisão (até esse ponto)

A herança (até esse ponto) ajuda:

- A evitar a duplicação de código
- A reutilizar o código
- A facilitar a manutenção
- Extensibilidade

## (Novo) código-fonte de Database

```
public class Database {
    private ArrayList<Item> items;

    /**
     * Cria uma classe Database vazia.
     */
    public Database()
    {
        items = new ArrayList<Item>();
    }

    /**
     * Adiciona um item ao banco de dados
     */
    public void addItem(Item theItem)
    {
        Items.add(theItem);
    }
    ...
}
```

## (Novo) código-fonte de Database

```
...  
/**  
 * Imprime uma lista de todos os CDs e DVDs  
 * armazenados atualmente no terminal de texto.  
 */  
public void list() {  
    for (Item item : items) {  
        items.print();  
        //Imprime uma linha em branco entre itens  
        System.out.println();  
    }  
}
```

## Subtipagem

- Primeiro, tínhamos:  

```
public void addCD(CD theCD)  
public void addDVD(DVD theDVD)
```
- Agora, temos:  

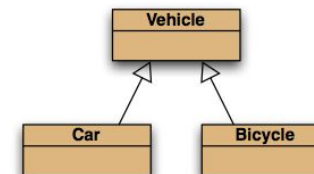
```
public void addItem(Item theItem)
```
- Chamamos esse método com:  

```
DVD myDVD = new DVD(...);  
database.addItem(myDVD);
```

## Subclasses e subtipos

- Classes definem tipos
- Subclasses definem subtipos
- Objetos de subclasses podem ser usados onde os objetos do supertipo são necessários
- Isso é chamado de **substituição**

## Subtipagem e atribuição



*Os objetos de subclasse podem ser atribuídos a variáveis da superclasse*

```
Vehicle v1 = new Vehicle();  
Vehicle v2 = new Car();  
Vehicle v3 = new Bicycle();
```

## Subtipagem e transmissão de parâmetro

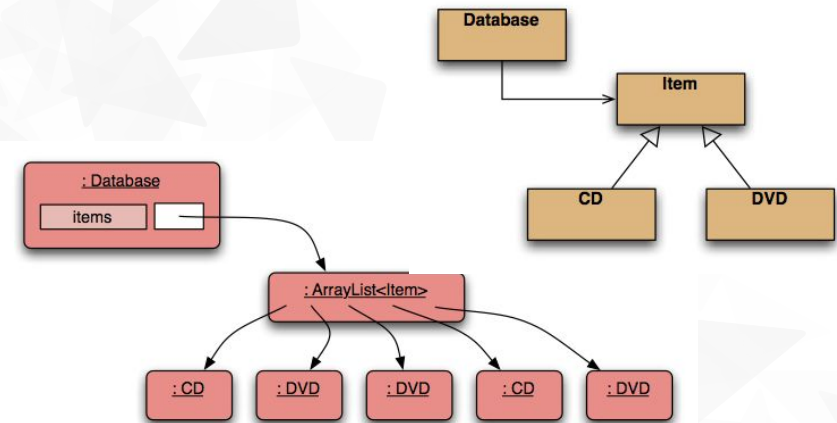
```
public class Database
{
    public void addItem(Item theItem)
    {
        ...
    }
}

DVD dvd = new DVD(...);
CD cd = new CD(...);

database.addItem(dvd);
database.addItem(cd);
```

*Os objetos de subclasse podem ser atribuídos a variáveis da superclasse*

## Diagramas de classes/objetos



## Variáveis polimórficas

## Variáveis polimórficas

- Variáveis de objeto em Java são **polimórficas**
- Podem armazenar objetos de mais de um tipo
  - Objetos do tipo declarado
  - ou
  - Objetos do subtipo do tipo declarado



## Coerção

- É possível atribuir o subtipo ao supertipo
- Não é possível atribuir o supertipo ao subtipo!!!

```
Vehicle v;  
Car c = new Car();  
v = c; //correto  
c = v; //erro em tempo de compilação
```

- A coerção corrige isso:

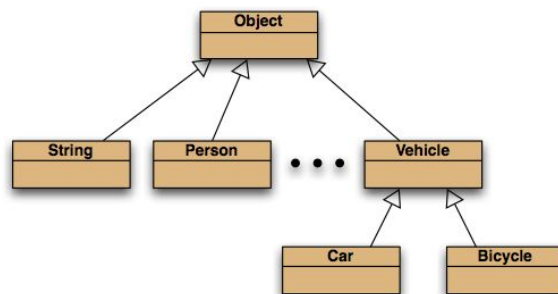
```
c = (Car)v;
```

OK apenas se o veículo for de fato um carro!

## Coerção

- Um tipo de objeto entre parênteses
  - Similar ao cast do C++
- Usado para superar 'perda do tipo'
- O objeto não é alterado
- Uma verificação em tempo de execução é feita para garantir que o objeto realmente é desse tipo
  - **ClassCastException** se não for!
- Utilize-a com moderação

## A classe Object



*Todas as  
classes herdam  
de **Object***

## Coleções polimórficas

- Todas as coleções são polimórficas
- Elementos são do tipo Object
  - “Lembra do ponteiro para nada (void \* variavel) de C++?”

```
public void add(Object element)  
public Object get(int index)
```

## Coleções e tipos primitivos

- Todos os objetos podem ser inseridos em coleções ...
- ... porque coleções aceitam elementos do tipo Object ...
- ... e todas as classes são subtipos de Object.
- E os tipos simples?

## Classes wrappers

- Tipos primitivos (int, char, etc) não são objetos
  - Precisam ser empacotados em um objeto
- Existem classes empacotadoras para todos os tipos simples

Tipo simples	Classe empacotadora
int	Integer
float	Float
char	Character
...	...

## Classes wrappers

```
int i = 18;  
Integer iWrap = new Integer(i);  
...  
int value = iWrap.intValue();
```

← empacota o valor

← desempacota o valor

*Na prática, autoboxing e unboxing significam que não precisamos fazer isso com frequência*

## Autoboxing e unboxing

- Autoboxing

```
private ArrayList<Integer> markList;  
...  
public void storeMark(int mark) {  
    markList.add(mark);  
}
```

- Unboxing

```
int firstMark = markList.remove(0);
```

## Revisão

- A herança permite a definição de classes como extensões de outras classes
- Herança
  - Evita duplicação de código
  - Permite reuso de código
  - Simplifica o código
  - Simplifica a manutenção e extensão
- Variáveis podem armazenar objetos de subtipo
- Subtipos podem ser usados sempre que se esperar objetos de supertipo (substituição)

Por hoje é só....

- Capítulo 8 do livro

## IMD0040 Linguagem de Programação 2

- Aula 08
- Aperfeiçoando estruturas com o uso da herança