Programación 2

TUDAI

Interfaces

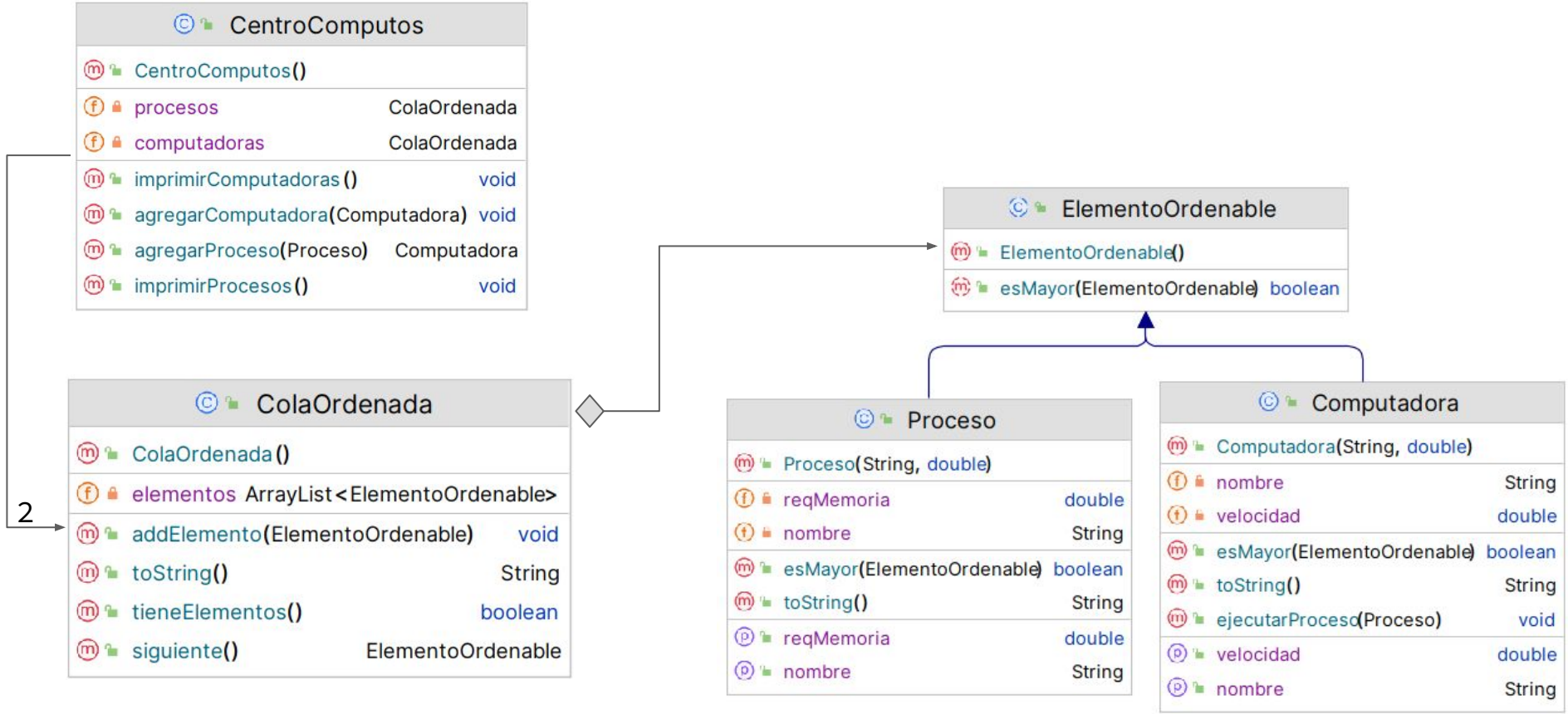
Centro de Cómputos (ej. 2, TP 6)

Un centro de cómputos se encarga de ejecutar procesos utilizando algunas de las computadoras que dispone.

Si no hay computadoras disponibles los procesos a ejecutar deben esperar en una cola de espera que los ordena teniendo en cuenta sus requerimientos de memoria (los procesos con mayor requerimiento de memoria serán atendidos en primer lugar).

Las computadoras disponibles para ejecutar procesos se ordenan en una cola que prioriza la selección de las computadoras más rápidas.

Centro de Cómputos (ej. 2, TP 6)

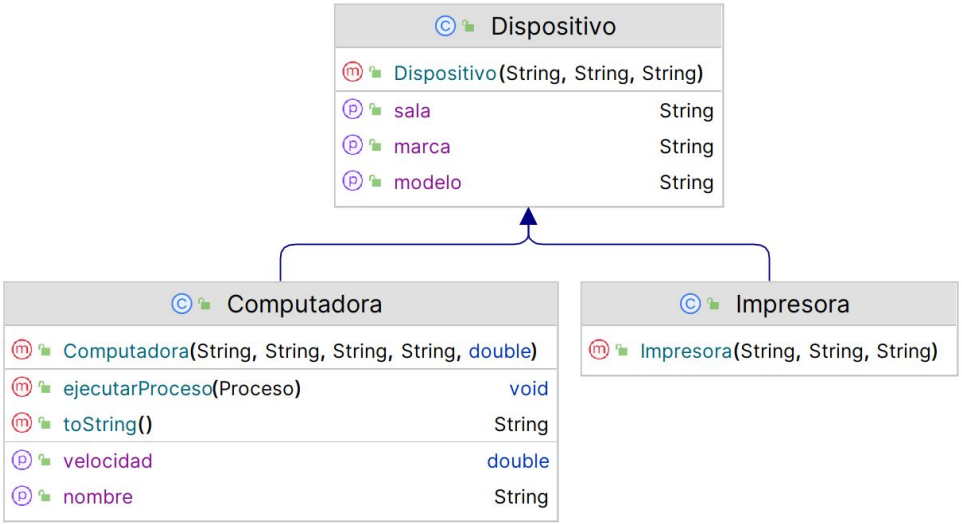


Centro de Cómputos (ej. 2, TP 6)

¿Que pasa si el sistema desea modelar también *impresoras*, que comparten información (atributos) y/o comportamiento (métodos) con las *computadoras*?

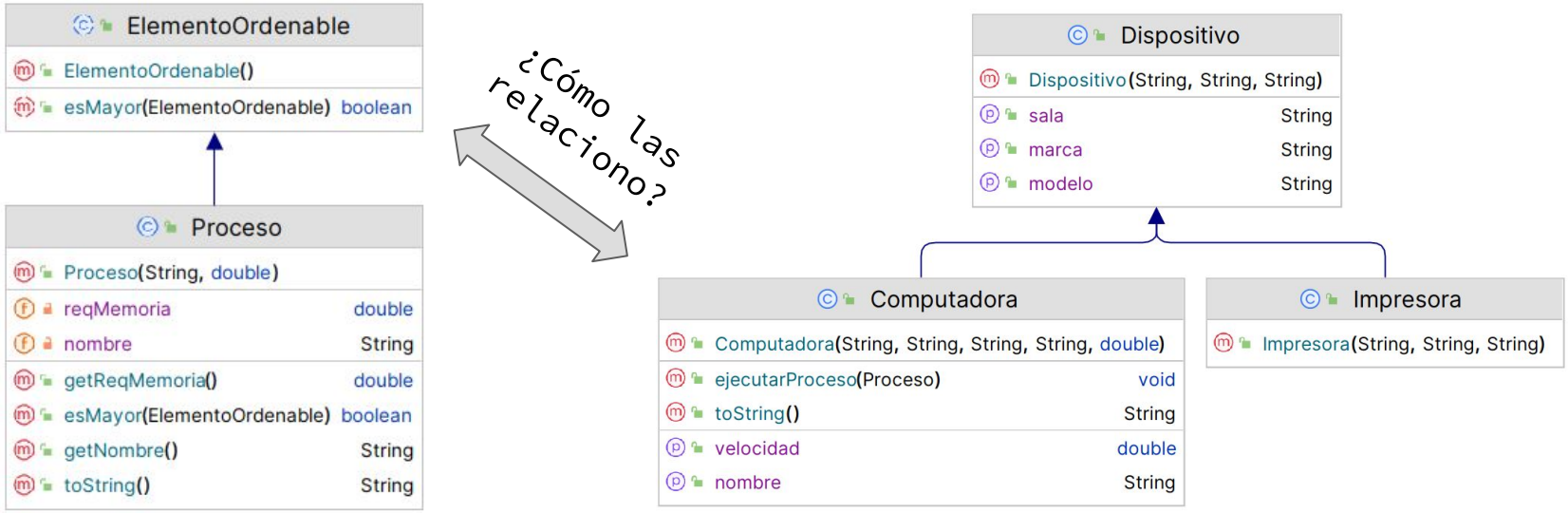
Centro de Cómputos (ej. 2, TP 6)

¿Que hubiera pasado si el sistema también modelaba *impresoras*, que comparten información (atributos) y/o comportamiento (métodos) con las *computadoras*?|| |



Centro de Cómputos (ej. 2, TP 6)

¿Que hubiera pasado si el sistema también modelaba *impresoras*, que comparten información (atributos) y/o comportamiento (métodos) con las *computadoras*?



Interfaces

- Java no permite herencia múltiple.

- Se incorpora el concepto de **Interface** para afrontar este tipo de problemas en los que, clases posiblemente no relacionadas entre sí, comparten parte de su ***comportamiento.***

Interfaces

- El concepto de interface lleva un paso más allá el concepto de clase abstracta

- Definen un protocolo de comportamiento, independientemente de dónde vaya a ser utilizado y proporcionan un formato común para implementarlo en las clases

Interfaces

- Ventaja
  - Desacople entre la definición del comportamiento y la clase que lo implementa

- Características conceptuales
  - Sólo tienen la signatura de los métodos
  - No implementan ningún método
  - No pueden tener atributos (solo constantes)

Interfaces

- Sintácticamente son similares a las clases pero se declaran con la palabra clave **interface**

```
public interface MiInterface{
    …
}
```

- Los métodos se declaran sin cuerpo y puede omitirse el modificador de acceso

```
void miMetodo();
```

- Todos los métodos de una interface son implícitamente public y abstract

Interfaces

- Todas las variables de una interface son implícitamente constantes (public static final), y esto se puede omitir en su declaración.

  ```
  int CANT_MAX = 100;
  ```

  Es constante, no variable

- Al ser constantes, deben incluir un valor inicial.

Interfaces

- Una interface puede **heredar** (extends) de *una o más* interfaces

```
public interface miInterface extends otraInterface, unaMas{
   …
}
```

- Una interface no puede heredar de otro elemento que no sea una interface

Interfaces

- Una **clase** puede indicar que *implementa* una **interface** mediante la palabra reservada `implements`.

  ```
  public class miClase implements miInterface
  ```

- Una interface no puede implementar otra interface. Para esto se usa herencia de interfaces (extends)

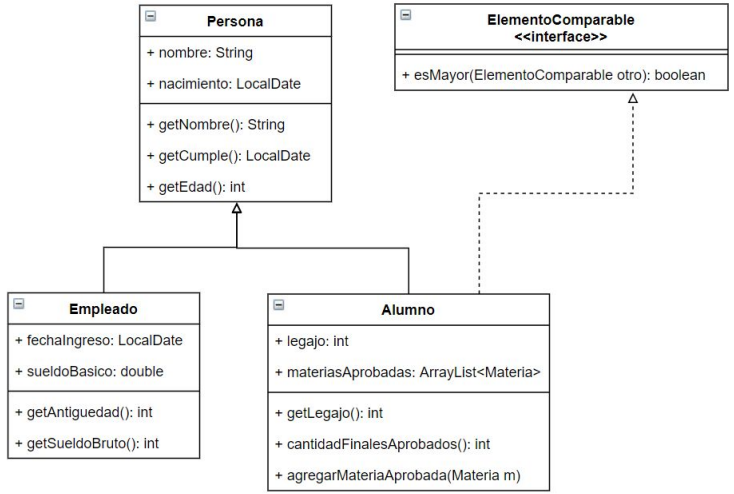
- Una interface puede ser implementada por cualquier cantidad de clases, y una clase puede implementar más de una interface

Interfaces

- Las clases que implementan una interface deben proporcionar comportamiento para todos los métodos definidos en la misma. Caso contrario, la clase debe declararse abstracta

Interfaces

- Los **tipos** de las interfaces pueden utilizados **polimórficamente**. Esto implica que pueden declararse variables o atributos del tipo de una interface




```
ElementoComparable a1 = new Alumno(“Juan Perez”, cumple1, 23779);
Alumno a2 = new Alumno(“Maria Garcia”, cumple2, 23780);

System.out.println( a1.esMayor(a2) );
System.out.println( a2.getNombre() );
System.out.println( ((Alumno) a1).getNombre() );
```

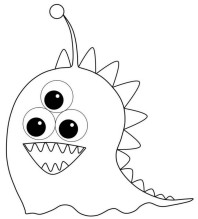
Interfaces

¿Cuándo uso clase abstracta y cuándo Interface?

- Usamos una clase abstracta cuando se necesita definir una plantilla para un grupo de subclases

- Usamos una interface cuando se debe definir un rol para otras clases, independientemente del árbol de herencia de estas clases

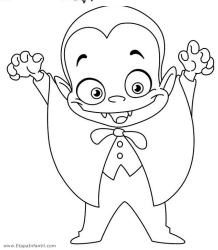
Ejemplo



```
interface Monstruo {
   void asustar();
}
```

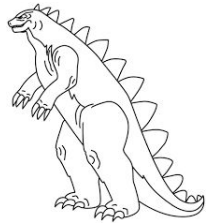
```
interface MonstruoPeligroso
     extends Monstruo {
   void destruir();
}
```

```
interface MonstruoLetal
   void matar();
}
```



```
interface Vampiro extends
     MonstruoPeligroso,
     MonstruoLetal {
   void chuparSangre();
}
```

```
class GodZilla implements MonstruoPeligroso {
   public void asustar() {
     System.out.println("Grrrrr");
   }
   public void destruir() {
     System.out.println("plaf!");
   }
}
```

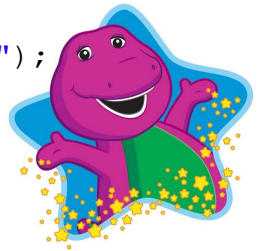


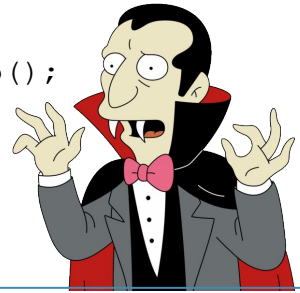
```
class VampiroMaléfico implements Vampiro {
   public void asustar() {
     System.out.println("buuuh!");
   }
   public void destruir() {
     System.out.println("boing!");
   }
   public void matar() {
     System.out.println("pum!");
   }
   public void chuparSangre() {
     System.out.println("fffffhhhhh");
   }
}
```

Ejemplo

```
public class HorrorShow{

   public HorrorShow(){}

   public void asusta(Monstruo m){
      m.asustar();
   }


   public void liquida(MonstruoLetal l){
      l.matar();
   }

   public void asustaMas(MonstruoPeligroso m){
      m.asustar();
      m.destruir();
   }
```

```
public static void main(String[] args) {
   HorrorShow show = new HorrorShow();
   MonstruoPeligroso sullivan = new MonstruoPeligroso();
   MonstruoPeligroso barney = new GodZilla();

   System.out.println("Sale barney");
   show.asusta(barney);
   show.asustaMas(barney);
   show.liquida(barney);

   System.out.println("Sale drácula");
   Vampiro dracula = new VampiroMaléfico();
   show.asusta(dracula);
   show.asustaMas(dracula);
   show.liquida(dracula);
}
```





Ejemplo

```
public class HorrorShow{

   public HorrorShow(){}

   public void asusta(Monstruo m){
      m.asustar();
   }


   public void liquida(MonstruoLetal l){
      l.matar();
   }

   public void asustaMas(MonstruoPeligroso m){
      m.asustar();
      m.destruir();
   }
```

```
public static void main(String[] args) {
   HorrorShow show = new HorrorShow();
   MonstruoPeligroso sullivan = new MonstruoPeligroso();
   MonstruoPeligroso barney = new GodZilla();

   System.out.println("Sale barney");
   show.asusta(barney);
   show.asustaMas(barney);
   show.liquida(barney);

   System.out.println("Sale drácula");
   Vampiro dracula = new VampiroMaléfico();
   show.asusta(dracula);
   show.asustaMas(dracula);
   show.liquida(dracula);
}
```

New de una interface!

Ejemplo

```
public class HorrorShow{

   public HorrorShow(){}

   public void asusta(Monstruo m){
      m.asustar();
   }


   public void liquida(MonstruoLetal l){
      l.matar();
   }

   public void asustaMas(MonstruoPeligroso m){
      m.asustar();
      m.destruir();
   }
```

```
public static void main(String[] args) {
   HorrorShow show = new HorrorShow();
   MonstruoPeligroso sullivan = new MonstruoPeligroso();
   MonstruoPeligroso barney = new GodZilla();

   System.out.println("Sale barney");
   show.asusta(barney);
   show.asustaMas(barney);
   show.liquida(barney);

   System.out.println("Sale drácula");
   Vampiro dracula = new VampiroMaléfico();
   show.asusta(dracula);
   show.asustaMas(dracula);
   show.liquida(dracula);
}
```

Ejemplo

```
public class HorrorShow{

   public HorrorShow(){}

   public void asusta(Monstruo m){
      m.asustar();
   }


   public void liquida(MonstruoLetal l){
      l.matar();
   }

   public void asustaMas(MonstruoPeligroso m){
      m.asustar();
      m.destruir();
   }
```

```
public static void main(String[] args) {
   HorrorShow show = new HorrorShow();
   MonstruoPeligroso sullivan = new MonstruoPeligroso();
   MonstruoPeligroso barney = new GodZilla();

   System.out.println("Sale barney")
   show.asusta(barney);
   show.asustaMas(barney);
   show.liquida(barney);

   System.out.println("Sale drácula");
   Vampiro dracula = new VampiroMaléfico();
   show.asusta(dracula);
   show.asustaMas(dracula);
   show.liquida(dracula);
}
```

"Grrrrr"
"Plaf!"

Ejemplo

```
public class HorrorShow{

   public HorrorShow(){}

   public void asusta(Monstruo m){
      m.asustar();
   }


   public void liquida(MonstruoLetal l){
      l.matar();
   }

   public void asustaMas(MonstruoPeligroso m){
      m.asustar();
      m.destruir();
   }
```

```
public static void main(String[] args) {
   HorrorShow show = new HorrorShow();
   MonstruoPeligroso sullivan = new MonstruoPeligroso();
   MonstruoPeligroso barney = new GodZilla();

   System.out.println("Sale barney")
   show.asusta(barney);
   show.asustaMas(barney);
   show.liquida(barney);

   System.out.println("Sale drácula");
   Vampiro dracula = new VampiroMaléfico();
   show.asusta(dracula);
   show.asustaMas(dracula);
   show.liquida(dracula);
}
```

Barney no es un MonstruoLetal

Ejemplo

```
public class HorrorShow{

   public HorrorShow(){}

   public void asusta(Monstruo m){
      m.asustar();
   }


   public void liquida(MonstruoLetal l){
      l.matar();
   }

   public void asustaMas(MonstruoPeligroso m){
      m.asustar();
      m.destruir();
   }
```

```
public static void main(String[] args) {
   HorrorShow show = new HorrorShow();
   MonstruoPeligroso sullivan = new MonstruoPeligroso();
   MonstruoPeligroso barney = new GodZilla();

   System.out.println("Sale barney")
   show.asusta(barney);
   show.asustaMas(barney);
   show.liquida(barney);

   System.out.println("Sale drácula");
   Vampiro dracula = new VampiroMaléfico();
   show.asusta(dracula);
   show.asustaMas(dracula);
   show.liquida(dracula);
}
```

Ejemplo

```
public class HorrorShow{

   public HorrorShow(){}

   public void asusta(Monstruo m){
      m.asustar();
   }


   public void liquida(MonstruoLetal l){
      l.matar();
   }

   public void asustaMas(MonstruoPeligroso m){
      m.asustar();
      m.destruir();
   }
```

```
public static void main(String[] args) {
   HorrorShow show = new HorrorShow();
   MonstruoPeligroso sullivan = new MonstruoPeligroso();
   MonstruoPeligroso barney = new GodZilla();

   System.out.println("Sale barney")
   show.asusta(barney);
   show.asustaMas(barney);
   show.liquida(barney);

   System.out.println("Sale drácula");
   Vampiro dracula = new VampiroMaléfico();
   show.asusta(dracula);
   show.asustaMas(dracula);
   show.liquida(dracula);
}
```

"buuuh!"
"boing!"

Ejemplo

```
public class HorrorShow{

   public HorrorShow(){}

   public void asusta(Monstruo m){
      m.asustar();
   }


   public void liquida(MonstruoLetal l){
      l.matar();
   }

   public void asustaMas(MonstruoPeligroso m){
      m.asustar();
      m.destruir();
   }
```

```
public static void main(String[] args) {
   HorrorShow show = new HorrorShow();
   MonstruoPeligroso sullivan = new MonstruoPeligroso();
   MonstruoPeligroso barney = new GodZilla();

   System.out.println("Sale barney")
   show.asusta(barney);
   show.asustaMas(barney);
   show.liquida(barney);

   System.out.println("Sale drácula");
   Vampiro dracula = new VampiroMaléfico();
   show.asusta(dracula);
   show.asustaMas(dracula);
   show.liquida(dracula);
}
```