

Programación 2

Tecnicatura en Desarrollo de Aplicaciones
Informáticas

Clase Abstracta



Clase Abstracta : Abstracción

Normalmente usamos las abstracciones para poder referirnos a algo enfocándonos en el aspecto que nos interesa y no en detalles innecesarios

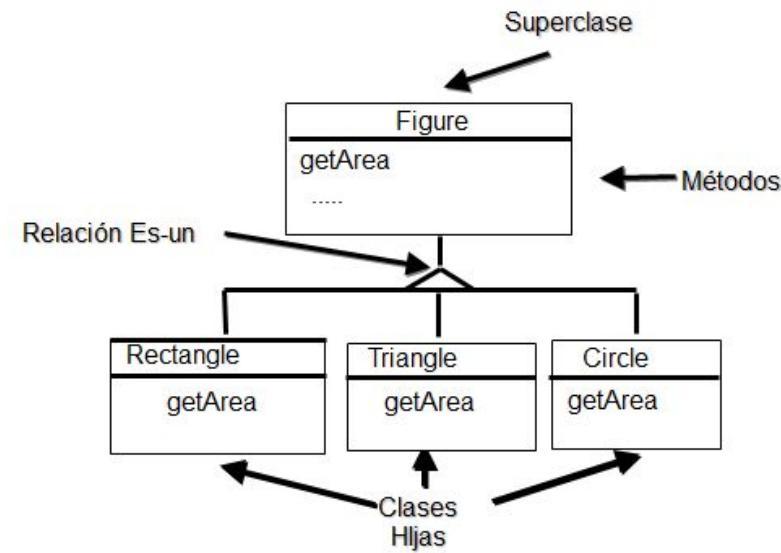
Ejemplo: el concepto de mueble, si digo “para mudarme tengo que llevar todos los muebles”, me enfoco en lo que quiero decir y no en el listado de cada mueble y sus detalles (tipo de mueble, forma, color, etc)

Clase Abstracta

En POO se pueden definir clases que representan un concepto abstracto

El concepto de Figura geométrica es una clase Abstracta

Los conceptos abstractos no pueden ser instanciados



Clase Abstracta

Puedo tener instancias de una Figura?

La respuesta es NO, porque como se calcula su área? cual es su perímetro? El concepto de figura me abstractea comportamiento común y características que deben tener las figuras.

El círculo es un elemento concreto al cual le puedo preguntar su area, su perímetro

Clase Abstracta

Una clase abstracta es una clase que puede ser extendida, pero no se pueden crear instancias (**no se le puede hacer new**)

Se usa la palabra clave **abstract** en la declaración

```
public abstract class Figura { . . . . }
```

Clase Abstracta

NO se puede crear una instancia de una clase abstracta, si se lo intenta se genera un error de compilación

```
Figura    ff = new Figura();
```



Clase Abstracta : Método abstracto

Una clase abstracta además de métodos (concretos) y atributos posee **métodos abstractos**

Un Método abstracto define comportamiento común de todos los objetos de las subclases concretas de la clase abstracta

```
public abstract class Figura {  
    public abstract double getArea();  
}
```

..

Clase Abstracta: Método Abstracto

Si la clase posee un método abstracto, **la clase es una clase abstracta** y por ende debe declararse como abstracta

Clase Abstracta: hijos

Una clase que hereda de una clase Abstracta debe implementar **TODOS** los métodos abstractos de la clase de la que hereda o debe declararse abstracta

```
public class Circulo extends Figura {  
  
    // Circulo Debe implementar todos los métodos abstractos de figura, getArea y  
    getPerimetro
```

Clase Abstracta: hijos

```
public abstract class FigAreaFija extends Figura {  
    double areaFija;  
  
    public double getArea(){ //Implementa getArea que era abstracto  
        return areaFija;  
    }  
    //Como no implementa getPerimetro, que tambien era abstracto debe declararse  
    abstracta la clase  
}
```

Clase Abstracta: hijos

```
public class FiguraFija extends FigAreaFija {  
    //Solo debe implementar un método abstracto getPerimetro, el otro ya lo implemento  
    FigAreaFija  
  
    double perimetro;  
  
    public double getPerimetro() {  
        return perimetro;  
    }  
}
```

Clases Abstractas: Cuidado

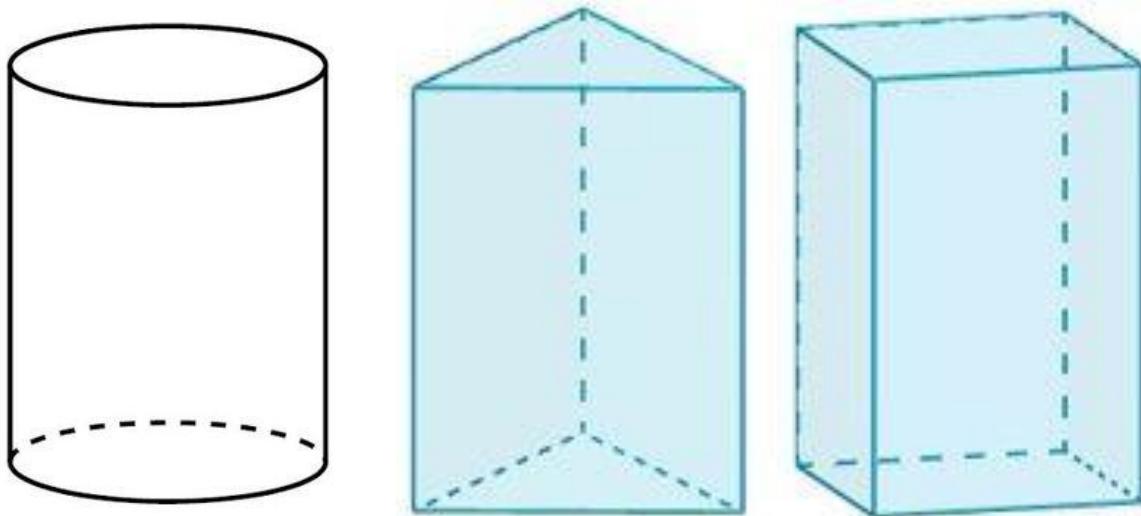
Conceptualmente en la Programación Orientada a Objetos **NO** deben existir clases abstractas que no tengan al menos un método abstracto.

Esto se aclara porque Java permite **clases abstractas sin métodos abstractos** (también deja poner atributos públicos.....)



Ejemplo Figuras3D

Figura3D



¿Qué cambia de la implementación realizada?

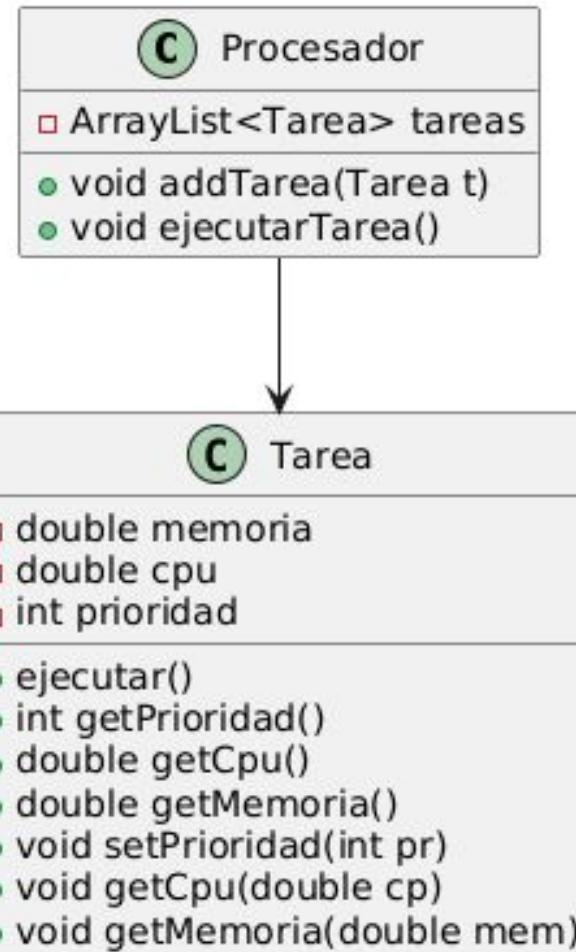
Ejemplo de Procesadores

Ejercicio

A un procesador se le asignan tareas, las mismas poseen una prioridad, uso de memoria y uso de CPU. El procesador X ordena las tareas de acuerdo a la prioridad de las mismas, mientras que otros procesadores las ordenan por uso de CPU o uso de memoria.

Procesadores

Primer aproximación a la solución



Procesadores : Clase Processor

```
public class Procesador{  
    ArrayList<Tareas> tareas;  
    public Procesador() {  
        tareas = new ArrayList();  
    }  
  
    public void ejecutarTarea() {  
        if (tareas.size()>0) {  
            Tarea tar= tareas.get(0);  
            tareas.remove(0);  
            tar.ejecutar();  
        }  
    }  
}
```

Procesadores : Clase Processor 2 Parte

```
public void addTarea(Tarea task) {  
    int i = 0;  
    while ( (i<tareas.size()) &&  
            (tareas.getPrioridad() >  
(tareas.get(0).getPrioridad() ) )  
        ) {  
    i ++;  
}  
tareas.add(task, i);  
}  
SOLO ORDENA POR PRIORIDAD
```

Procesadores

El caso anterior solo modela por prioridad

¿Cómo permito que se ordene por el uso de memoria y el uso de procesador ?

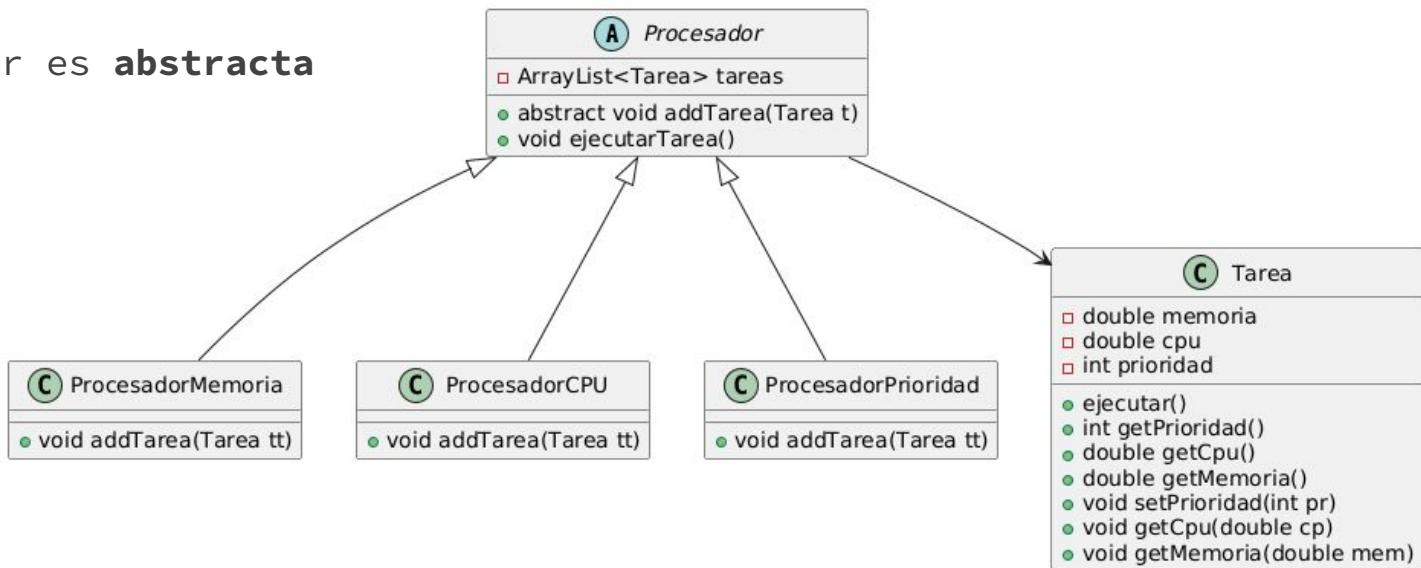


Procesador

Incorporo nuevos procesadores

agregarTarea es **abstracto** en Procesador

Entonces Procesador es **abstracta**



Procesador: add en ProcesadorMemoria

```
public void addTarea(Tarea tarea) {  
    int i = 0;  
    while ( (i < tareas.size()) &&  
        ( tarea.getMemoria() < (tareas.get(i).getMemoria() ) {  
        i++;  
    }  
    if (i < tareas.size())  
        tareas.add(tarea, i);  
    else  
        tareas.add(tarea);  
}
```

Procesador: add en ProcesadorCPU

```
public void addTarea(Tarea tarea) {  
    int i = 0;  
    while ( (i < tareas.size()) &&  
        ( tarea.getCpu() < (tareas.get(i).getCpu()) ) {  
        i++;  
    }  
    if (i < tareas.size())  
        tareas.add(tarea, i);  
    else  
        tareas.add(tarea);  
}
```

Procesadores: dudas

No son muy parecidos los dos códigos?

El procesador que ordena por prioridad, va a ser igual?

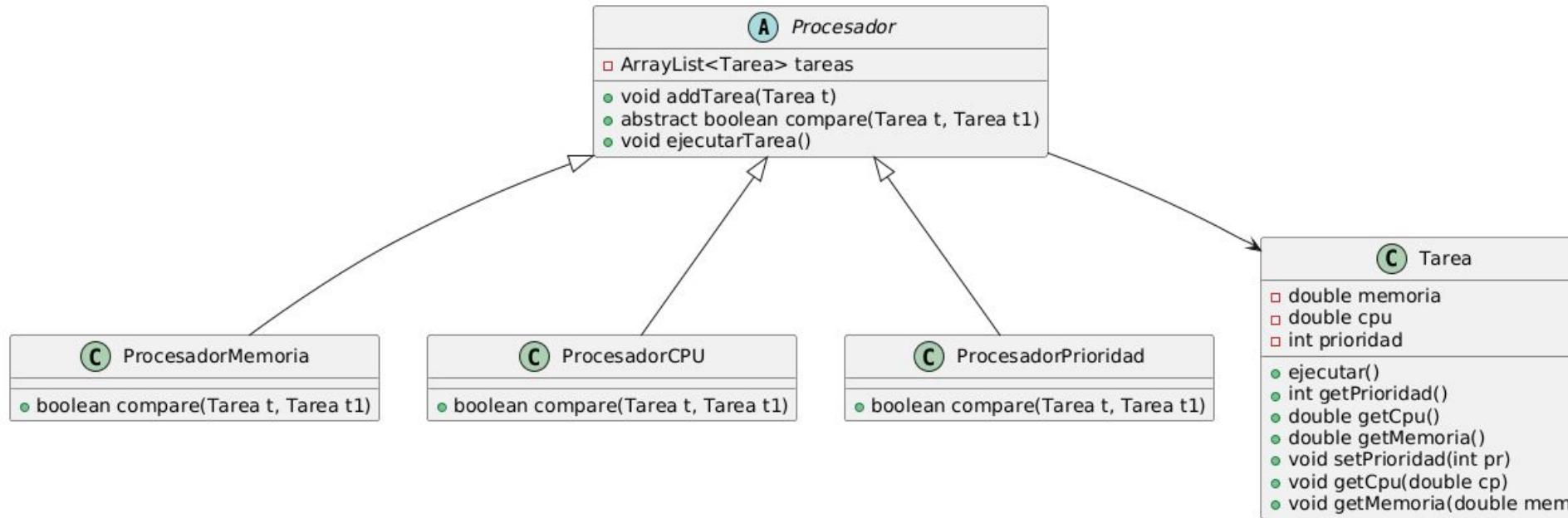


Procesadores: Abstracción del método add

En la clase Processor

```
public void addTarea(Tarea tarea) {  
    int i = 0;  
    while ( (i < tareas.size()) && ( this.compare (tarea, tareas.get(i)) ) ) {  
        i++;  
    }  
    if (i < tareas.size())  
        tareas.add(tarea,i);  
    else  
        tareas.add(tarea);  
}  
public abstract boolean compare(Tarea t1, Tarea t2); // en Processor no se como implementarlo
```

Procesador: Clases



Procesadores: Clase ProcesadorCpu

Debe implementar el método abstracto compare

```
public boolean compare(Tarea t1, Tarea t2){  
    return t1.getCpu() < t2.getCpu();  
}
```

Solo provee la comparación con el CPU de la tarea

Procesadores

ahora se quiere incorporar un procesador que agrega las tareas por orden de llegada!!!!

La tarea nueva siempre va al final de la lista de tareas!

Procesadores

Hay que preguntarse:

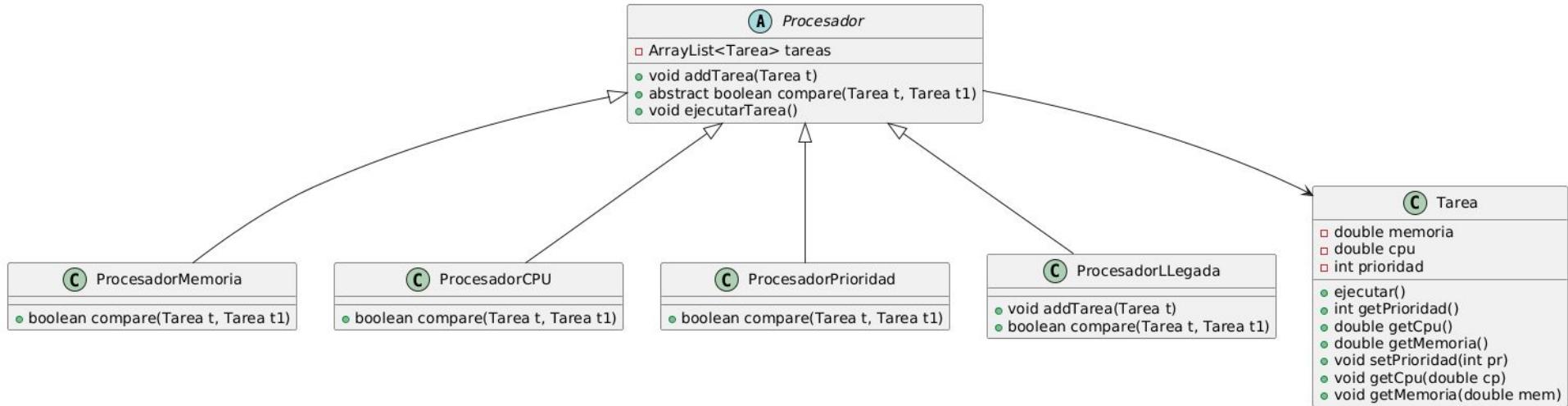
El procesador por orden de llegadas es un procesador?

SI

Tiene el mismo comportamiento que los demás?

SI, la misma interfaz (se le agregan tareas) pero cambia la forma en que las guarda

Procesadores



Procesadores: ProcesadorLLegada

```
public class ProcesadorLLegada extends Procesador{  
    public void addTarea (Tarea tarea){ //Redefine comportamiento  
        de la clase Procesador  
        tareas.add(tarea);  
    }  
  
    public boolean compare(Tarea t1, Tarea t2){  
        return false;  
    } // igual debe implementarlo porque es abstracto en  
    Procesador  
}
```

Si conocemos de antes que existe un procesador por orden de llegada la solución hubiese sido otra

