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

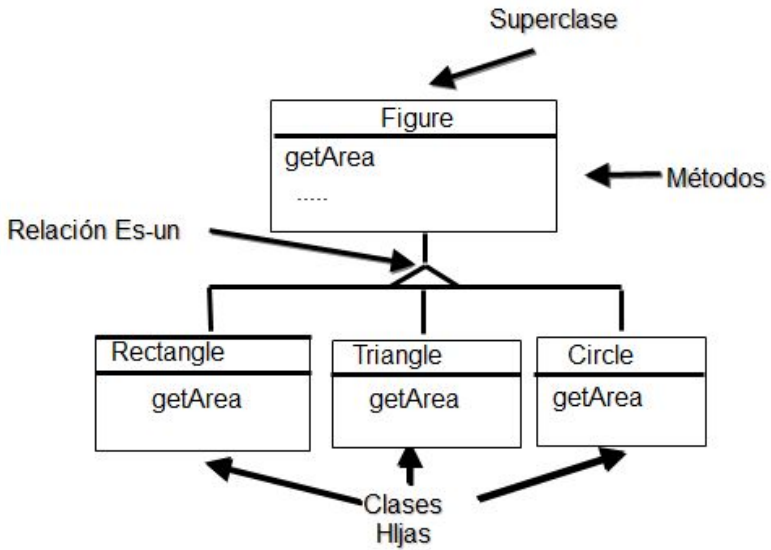
Ejemplo: el concepto de mueble, si digo “para mudarme tengo que llevar todos los muebles”, me enfoco en lo que quiero decir y no en el listado de cada mueble y sus detalles (tipo de mueble, forma, color, etc)

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 abstrae comportamiento común y características que deben tener las figuras.

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

Clase Abstracta

Una clase abstracta es una clase 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 {  . . . . }

public abstract class Formula{ . . . .}
```

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 Absstracta: 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 declarse
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;

    }

}
```

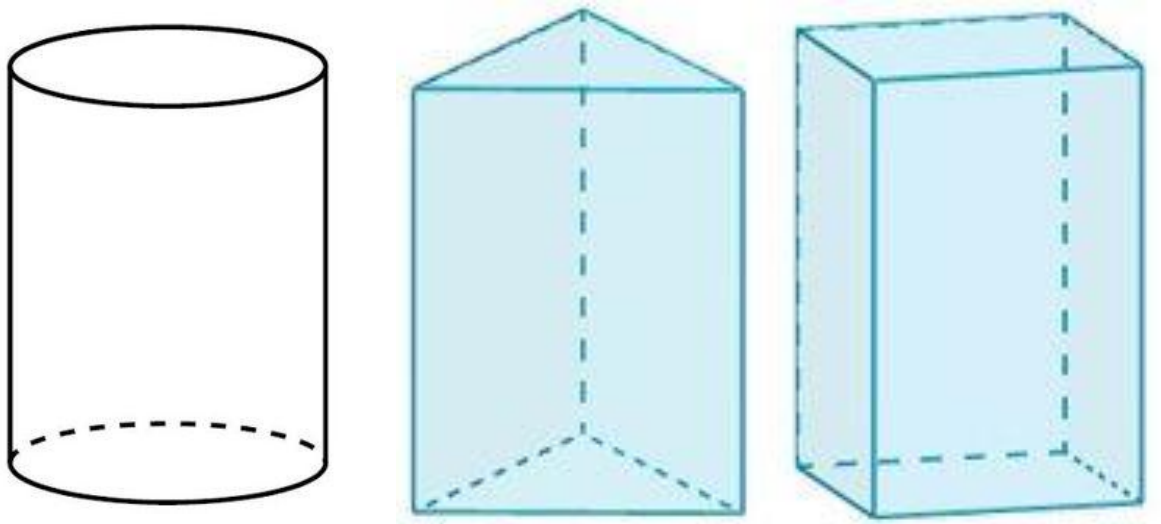
Clases Abstractas: Cuidado

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 publicos……)

Ejemplo Figuras3D

Figura3D



Simplificación en la cual debemos calcular el volumen de la figura

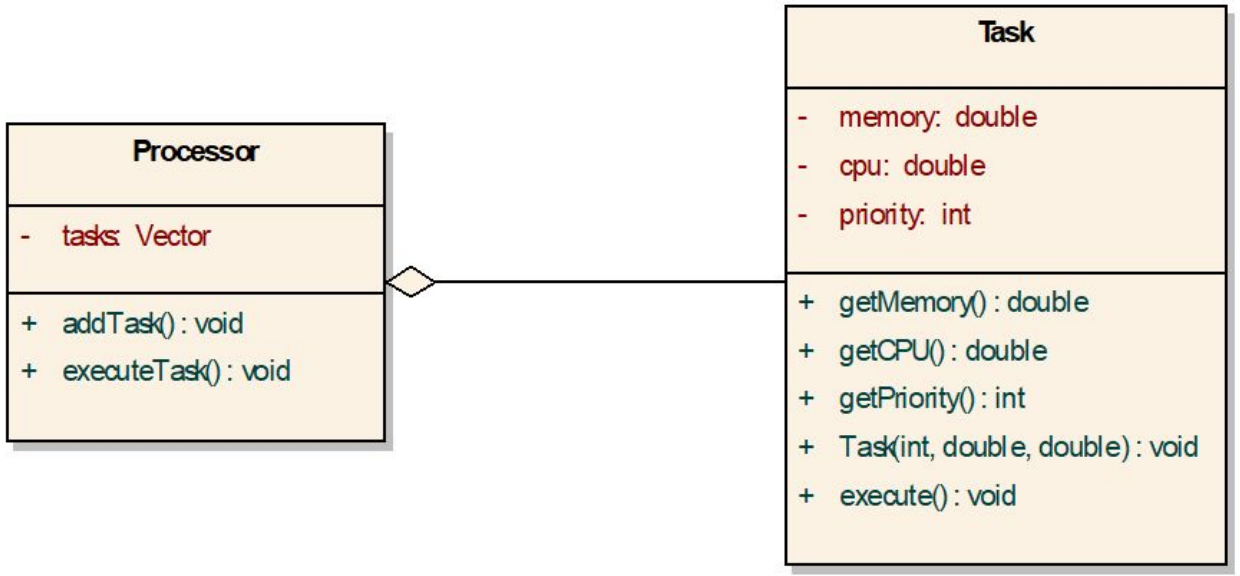
Ejemplo de Procesadores

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 Processor {
  Vector tasks;
  public Processor() {
     tasks = new Vector();
  }

  public void execute() {
    if (tasks.size()>0) {
      Task next = (Task)tasks.elementAt(0);
      tasks.removeElementAt(0);
      next.execute();
    }
```

Procesadores : Clase Processor 2 Parte

```
 ---
public void addTask(Task task) {
    int i = 0;
    while ( (i<tasks.size()) &&
                (task.getPriority() >
((Task)tasks.elementAt(i)).getPriority() )
              ) {
      i ++;
    }
tasks.insertElementAt(task, i);
}
SOLO ORDENA POR PRIORIDAD
```

Procesadores

El caso anterior solo modela por prioridad

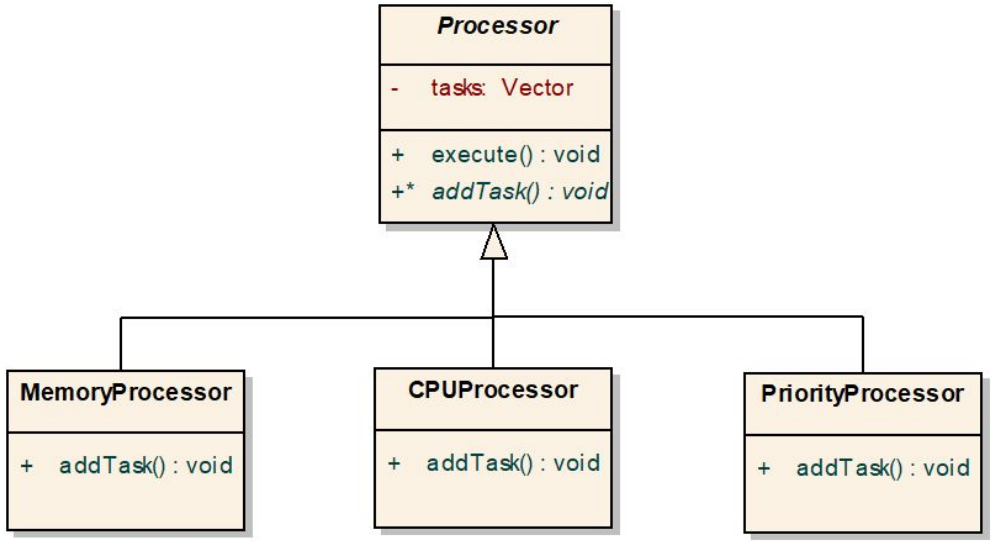
¿Cómo permito que se ordene por el uso de memoria y el uso de procesador ?

Procesador

Incorporo nuevos procesadores

add es **abstracto** en Processor

Entonces Processor es **abtracta**



Procesador: add en MemoryProcessor

– – –

```
public void addTask(Task tarea) {
      int i = 0;
      while ( (i < tasks.size()) &&
            ( tarea.getMemory()< ((Tarea)tasks.elementAt(i)).getMemory()  ) {
             i++;
      }
      if (i < tasks.size())
      tasks.insertElementAt(tarea, i);
      else
      tasks.add(tarea);
 }
```

Procesadores: add en CPUProcessor

– – –

```
public void addTask(Task tarea) {
        int i = 0;
        while ( (i < tasks.size()) &&
                ( tarea.getCPU()< ((Tarea)tasks.elementAt(i)).getCPU()  ) {
                 i++;
        }
        if (i < tasks.size())
        tasks.insertElementAt(tarea, i);
        else
        tasks.add(tarea);
 }
```

Procesadores: dudas

No son muy parecidos los dos códigos?

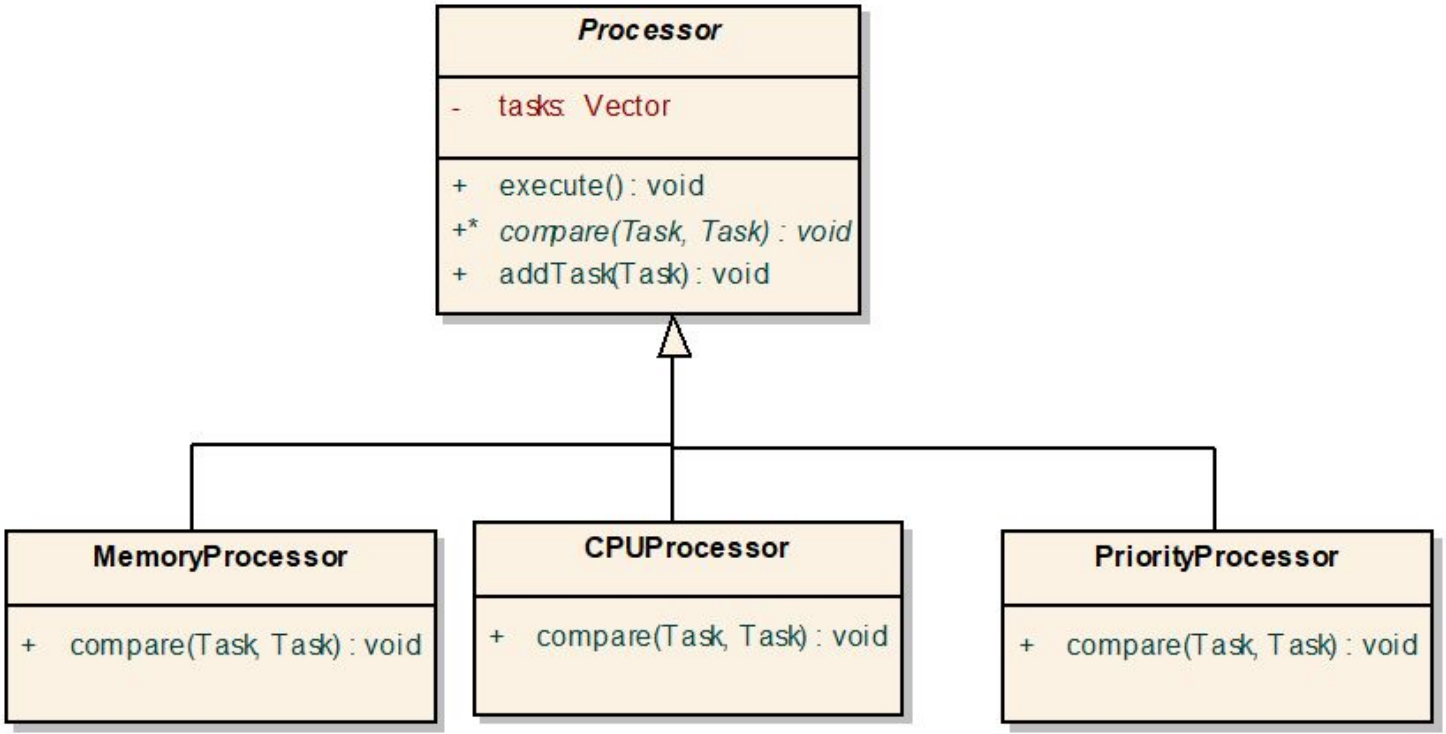
El procesador que ordena por prioridad, va a ser igual?

Procesadores: Abstracción del método add

En la clase Processor

```
public void addTask(Task tarea) {
        int i = 0;
        while ( (i < tasks.size()) &&  ( this.compare (tarea, ((Tarea)tasks.elementAt(i))  ) {
                i++;
        }
        if (i < tasks.size())
        tasks.insertElementAt(tarea, i);
        else
        tasks.add(tarea);
 }
public abstract boolean compare(Task  t1, Taskt2); // en Processor no se como implementarlo
```

Procesador: Clases



Procesadores: Clase CPUProcessor

Debe implementar el método abstracto compare

```
public boolean compare(Task t1, Task 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:

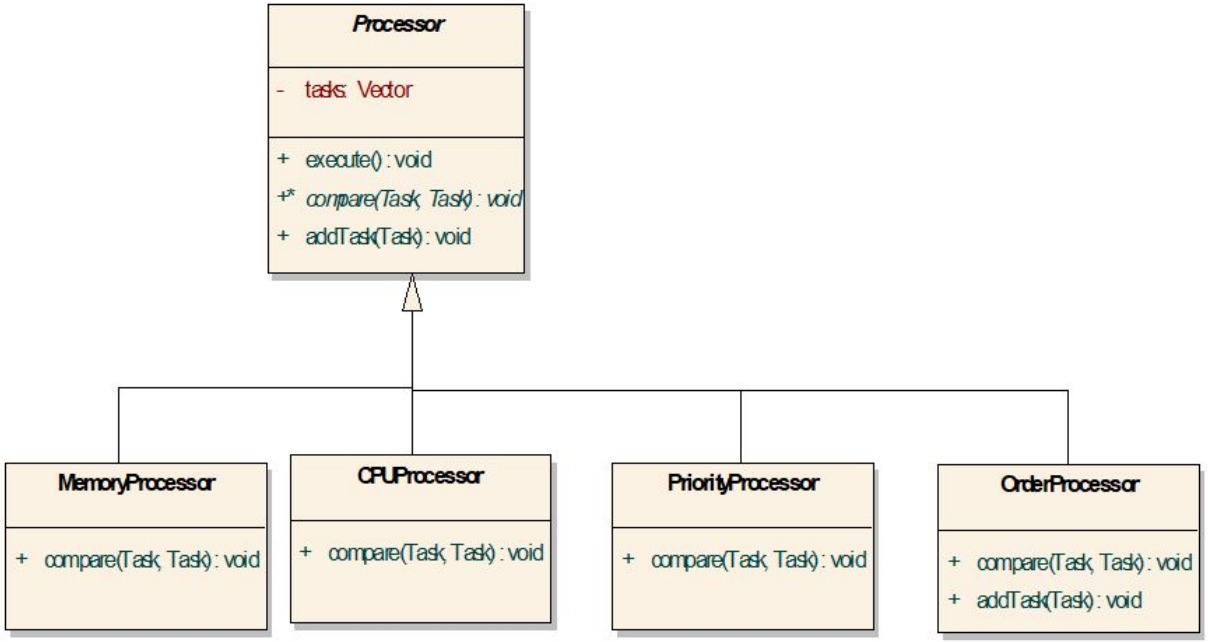
El procesador por orden de llegadas es un procesador?

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: OrderProcessor

```
public class OrderProcessor extends Processor {

    public void add (Task tarea){  //Redefine comportamiento de la
clase Processor
        taks.add(tarea);
     }

    public boolean compare(Task t1, Task t2){
       return false;
    } // igual debe implementarlo porque es abstracto en
Processor
}
```