Definición

* Ingeniería de Software:

Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software.

* Modelo:

Representación o especificación desde un punto de vista determinado y con un objetivo concreto

* Diseño de Software:

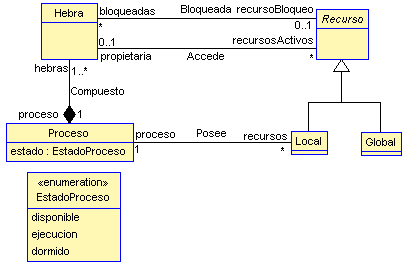
Conjuntos de planes y decisiones con los suficientes detalles como para permitir su realización física de acuerdo a unos requisitos.

* Patrón de diseño:

Solución probada que se puede aplicar con éxito a determinados problemas que aparecen repetidamente en el desarrollo y diseño de software

# Procesos, hebras y recursos

* 1. Para este sistema el diagrama de clases será el siguiente, obviando la existencia de una clase Sistema que lo englobe todo, ya que no se considera necesaria dado que en ningún momento se pide que se implementen varios sistemas distintos.



La relación bloqueada se creará cuando exista una propietaria, pero todo esto se controla a través de restricciones explícitas, indicadas en el siguiente apartado del ejercicio.

* 1. Hemos definido un total de 9 restricciones para este sistema, las restricciones son las siguientes:
* Proceso
  + inv maxHebrasProceso:

self.hebras->size() <= 32

* + inv maxProcesosSistema:

Proceso.allInstances->size() <= 128

* + inv dormidoEstaBloqueado:

self.estado = EstadoProceso::dormido implies

self.hebras.select(h|h.recursoBloqueo->notEmpty())->notEmpty()

* + inv soloUnoEjecucion:

Proceso.allInstances->select(p|p.estado = EstadoProceso::ejecucion)->size() <= 1

* Recurso
  + inv maxRecursosSistema:

Recurso.allInstances->size() <= 128

* + inv noBloqueadosSinPropietaria:

self.bloqueadas->notEmpty implies self.propietaria->notEmpty

* + inv propietarioNoBloqueado:

not self.bloqueadas->includes(self.propietaria)

* Hebra
  + inv maxHebrasSistema:

Hebra.allInstances->size() <= 64

* Local
  + inv noHebrasDeOtroProceso:

self.bloqueadas->including(self.propietaria)->forAll(h| h.proceso = self.proceso)

* Global
  + inv procesosEnEstadoCorrespondiente:

self.bloqueadas->forAll(h | h.proceso.estado = EstadoProceso::dormido)

* 1. Ni idea tbh

# Trabajadores y tareas

* 1. Para este apartado se nos pide traspasar a un DOO para implementar en Java el diagrama. Uno de los problemas que tenemos es la relación Contrato, que, para implementarla en Java habría que implementarla en una clase aparte haciendo que este se encargue de mantener la consistencia en la relación. También necesitamos que el atributo totalSalary sea derivado, pudiendo hacer que cada vez que introduzcamos o eliminemos un contrato nuevo en la relación se sume o reste el salario de éste al total. Para la relación entre Manager y Employee podemos hacer que las propias clases se encarguen de la consistencia sin introducir una nueva, ya que la multiplicidad de un extremo es 0..1.
  2. asd
  3. asd