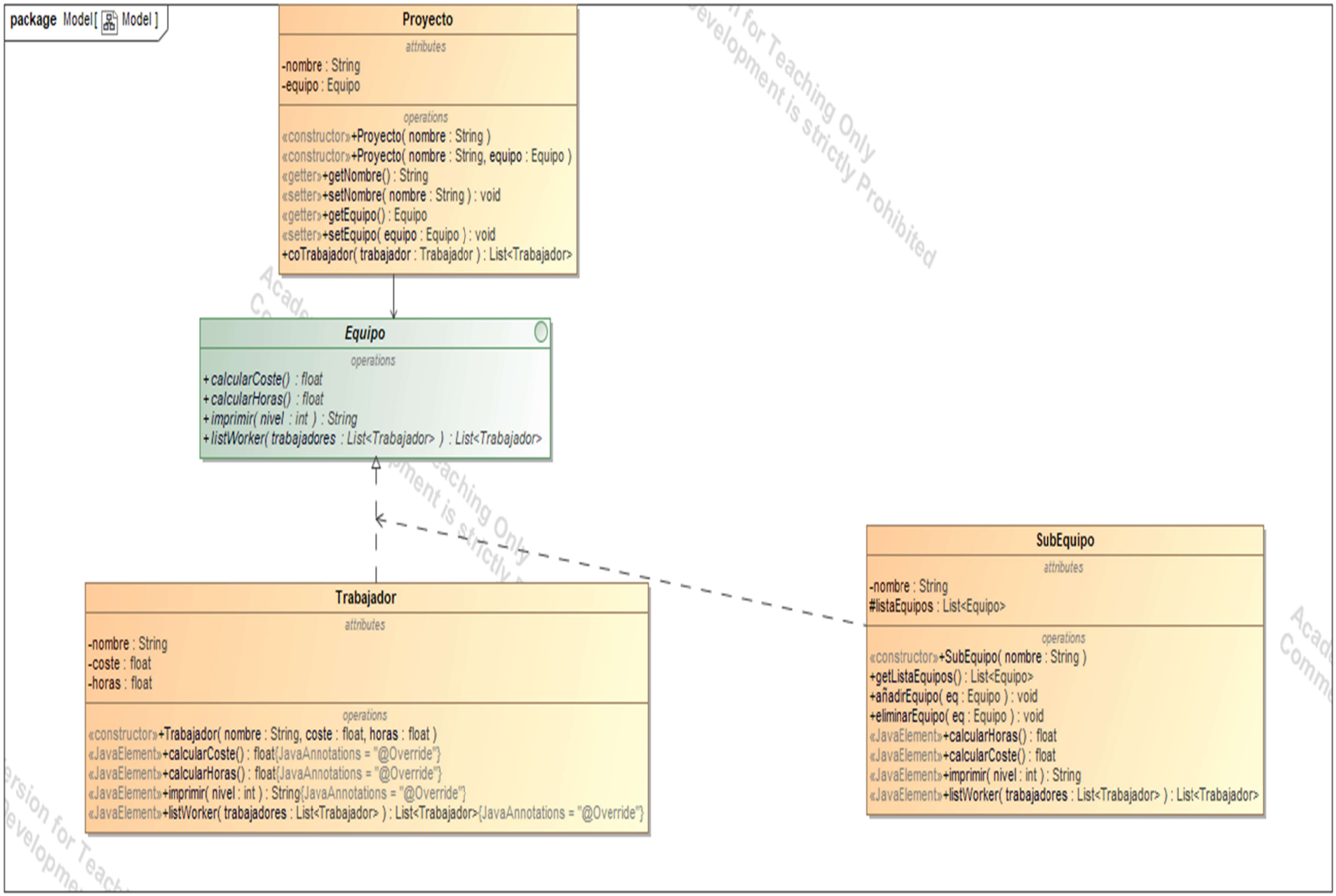
INFORME

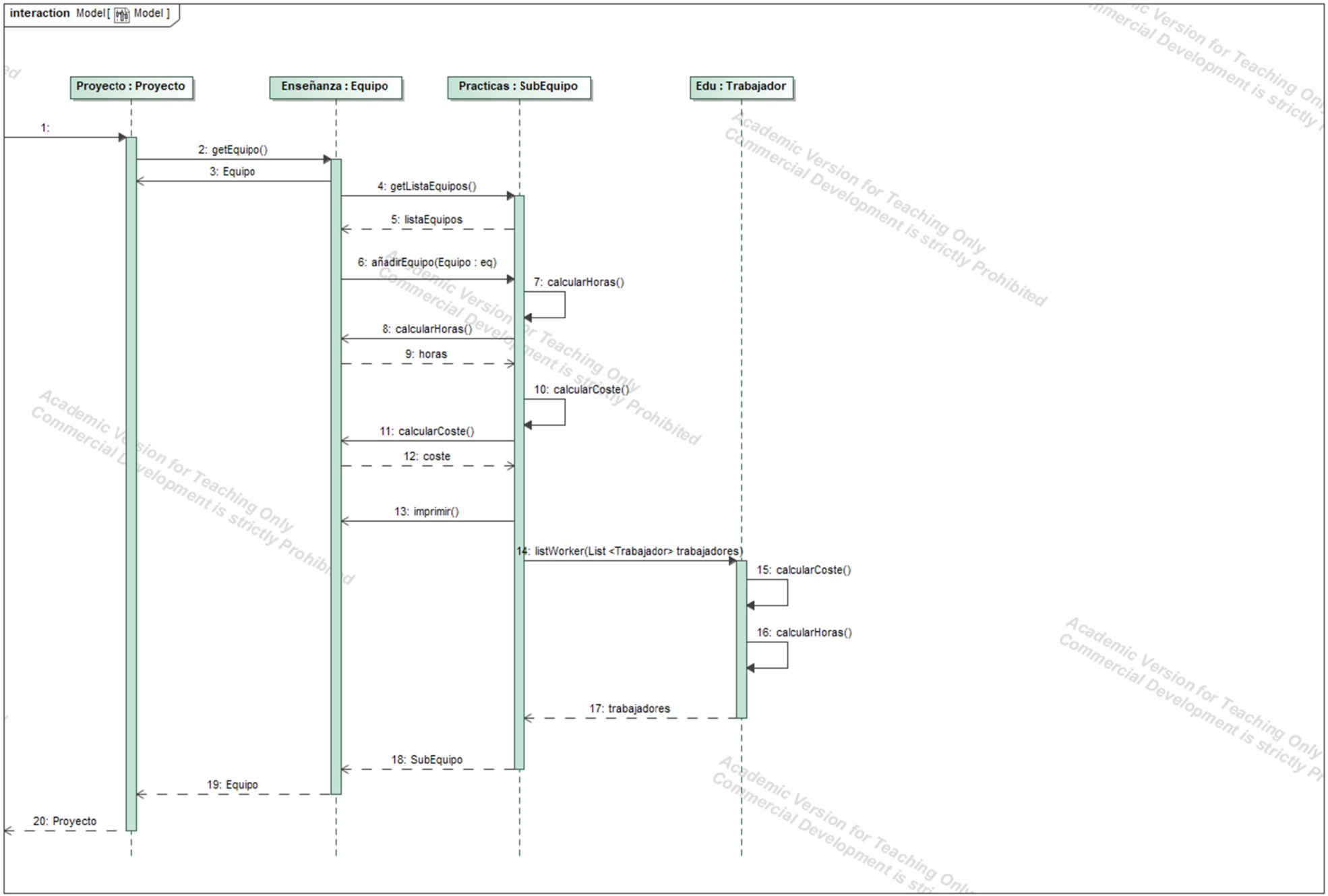
Ejercicio 1 – Control de la temperatura

- Patrón de diseño:
  - Patrón Estado: Este patrón permite a un objeto modificar su conducta al cambiar su estado interno.
    En este ejercicio, el interfaz EstadoTermostato encapsula los comportamientos de los estados del contexto, el cuál se define en la clase Termostato. Los distintos estados en los que se encuentra el termostato se definen en las clases Off, Manual, Timer y Program.
- Principio de diseño:
  - Principio de inversión de la dependencia: Es la estrategia de depender de interfaces o clases y funciones abstractas, en vez de depender de clases y funciones concretas.
    En este ejercicio, tanto la clase principal (Termostato) como los distintos estados, dependen del interfaz EstadoTermostato, por lo que la dependencia entre ellos se ha invertido.

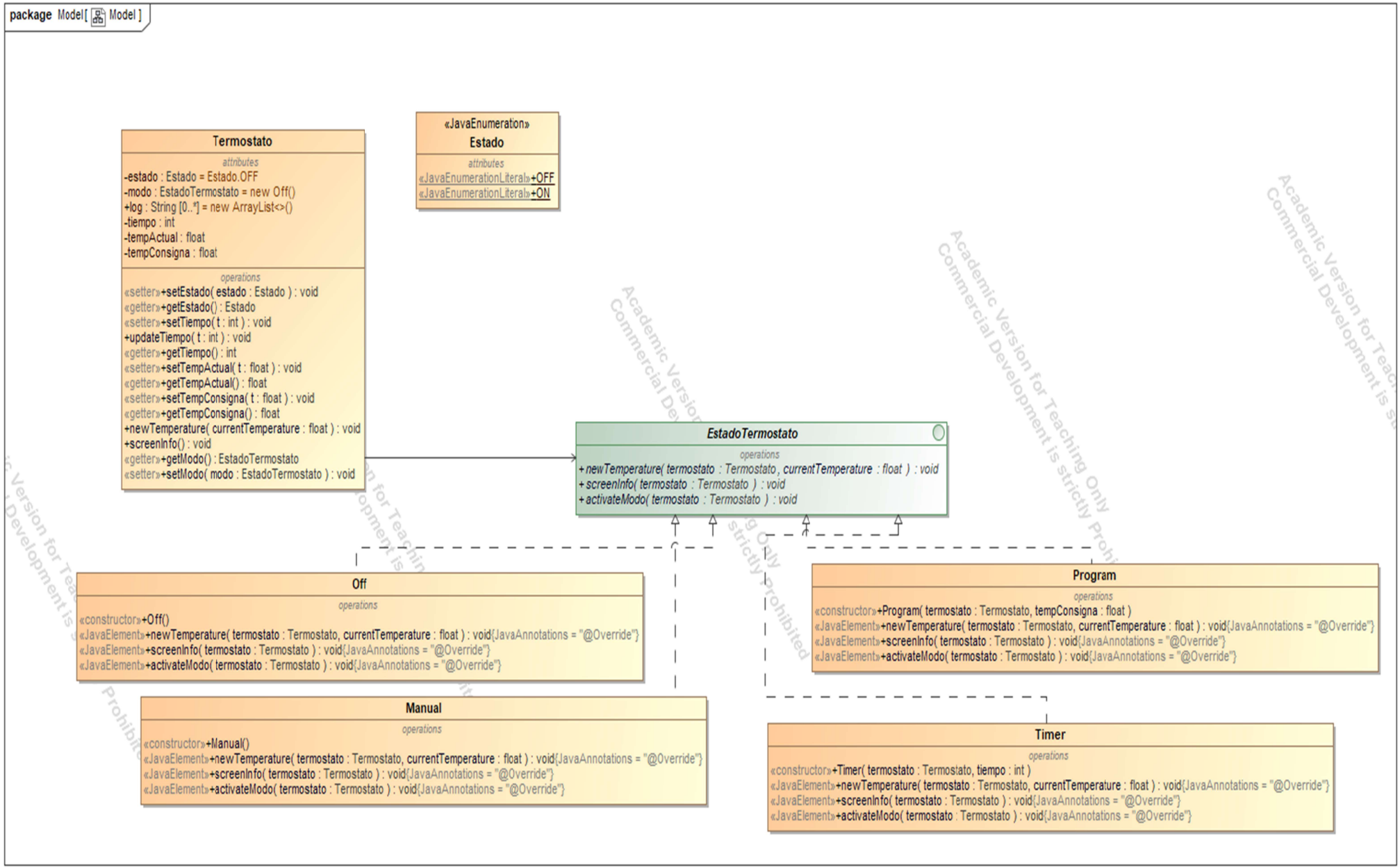
Ejercicio 2 – Gestión de equipos de trabajo

- Patrón de diseño:
  - Patrón Composición: Este patrón permite componer objetos en estructuras de árbol que representan jerarquías todo-parte
    En este ejercicio el componente es nuestro interfaz Equipo y la clase Proyecto es nuestra composición. La clase SubEquipo es nuestro componente concreto. Las ventajas de usar este patrón son que permite tratar a los objetos básicos y compuestos de manera uniforme y que facilita la introducción de nuevos componentes.
- Principio de diseño:
  - Principio de responsabilidad única: Cada objeto debe tener una responsabilidad única que esté enteramente encapsulada en la clase.
    En este ejercicio tenemos una clase Equipo en la cual un cambio en ella afecta a las clases no relacionadas con esa responsabilidad como por ejemplo la clase Trabajadores.
  - Principio abierto-cerrado: Las entidades software deberían ser abiertas para permitir su extensión, pero cerradas frente a la modificación.
    En este ejercicio si queremos cambiar el sueldo de un trabajador con el método calcularCoste(), podemos cambiarlo sin apenas modificar el código.

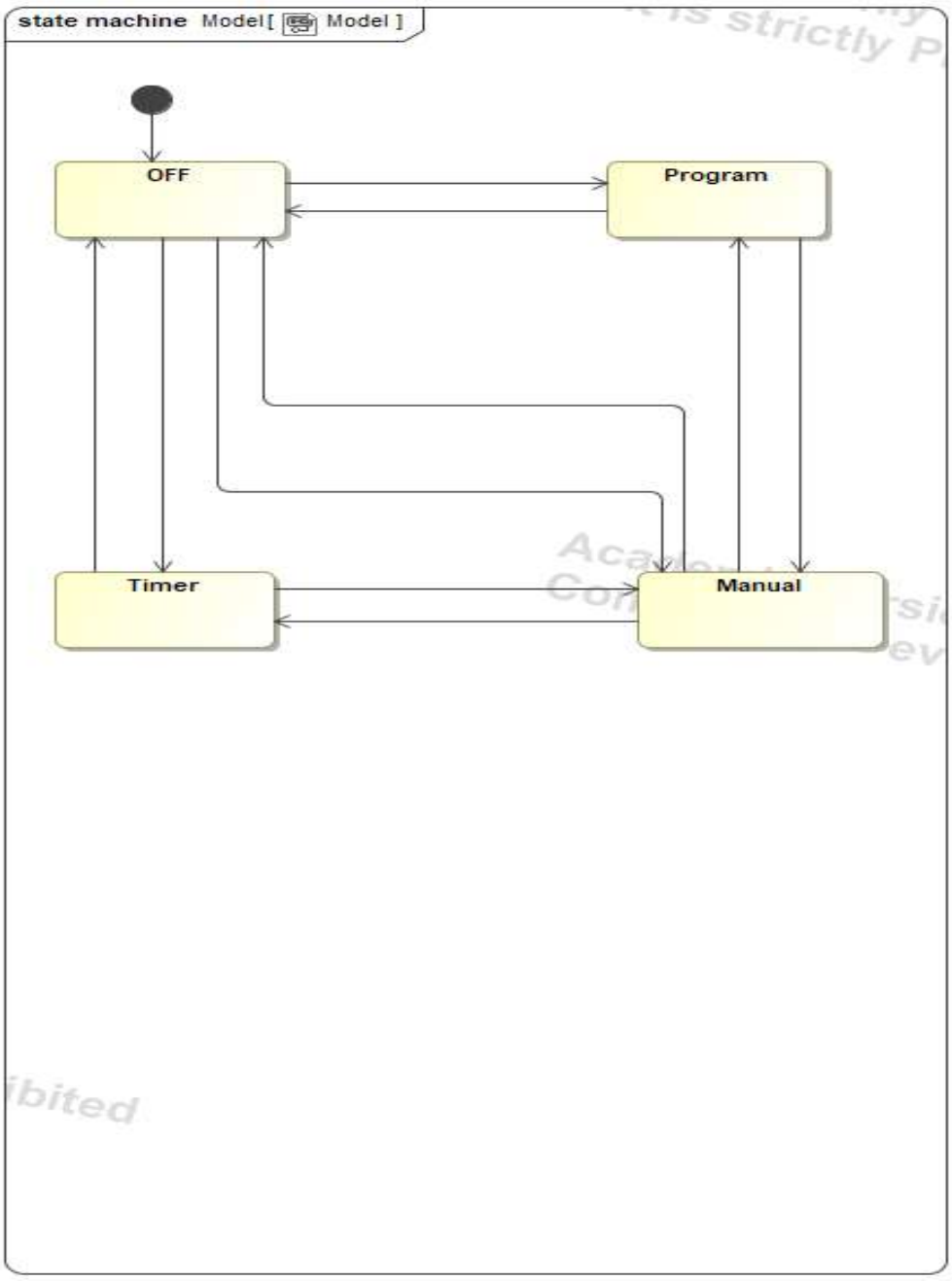

package Model[Model]
Proyecto
attributes
-nombre : String
-equipo : Equipo
operations
«constructor»+Proyecto(nombre : String)
«constructor»+Proyecto(nombre : String, equipo : Equipo)
«getter»+getNombre() : String
«setter»+setNombre(nombre : String) : void
«getter»+getEquipo() : Equipo
«setter»+setEquipo(equipo : Equipo) : void
+coTrabajador(trabajador : Trabajador) : List<Trabajador>
Equipo
operations
+calcularCoste() : float
+calcularHoras() : float
+imprimir(nivel : int) : String
+listWorker(trabajadores : List<Trabajador>) : List<Trabajador>
Trabajador
attributes
-nombre : String
-coste : float
-horas : float
operations
«constructor»+Trabajador(nombre : String, coste : float, horas : float)
«JavaElement»+calcularCoste() : float{JavaAnnotations = "@Override"}
«JavaElement»+calcularHoras() : float{JavaAnnotations = "@Override"}
«JavaElement»+imprimir(nivel : int) : String{JavaAnnotations = "@Override"}
«JavaElement»+listWorker(trabajadores : List<Trabajador>) : List<Trabajador>{JavaAnnotations = "@Override"}
SubEquipo
attributes
-nombre : String
#listaEquipos : List<Equipo>
operations
«constructor»+SubEquipo(nombre : String)
+getListaEquipos() : List<Equipo>
+añadirEquipo(eq : Equipo) : void
+eliminarEquipo(eq : Equipo) : void
«JavaElement»+calcularHoras() : float
«JavaElement»+calcularCoste() : float
«JavaElement»+imprimir(nivel : int) : String
«JavaElement»+listWorker(trabajadores : List<Trabajador>) : List<Trabajador>



Academic Version for Teaching Only
Commercial Development is strictly Prohibited



Academic Version for Teaching Only
Commercial Development is strictly Prohibited



Autores:

Fernando Seara Romera

Eduardo Pérez Fraguela