

Solución colaborativa de la entrega 1

Archivo deportes.wlk

```
object tenis {
  const costoPorMesPorTenista = 20000
  const costoPorEntrenadorPorTenista = 3000

  method presupuestoBase(unTenista){
    return costoPorMesPorTenista +
      costoPorEntrenadorPorTenista * unTenista.cantidadEntrenadores()
  }
}

object judo {

  const costoPorMesPorJudoka = 16000
  const costoPorLesionSparring = 2000

  method presupuestoBase(unJudoka){
    return costoPorMesPorJudoka +
      costoPorLesionSparring * unJudoka.sparringsLesionados()
  }
}
```

Archivo deportistas.wlk

```
import deportes.*

object delpo {
  var nombre = "Juan Martín Del Potro"
  var edad = 29
  var disciplina = tenis
  var cantidadRaquetas = 20
  const costoPorRaqueta = 5000
  var cantidadEntrenadores = 2

  method cantidadEntrenadores(){
    return cantidadEntrenadores
  }
}
```

```

method cantidadEntrenadores(unaCantidad){
    cantidadEntrenadores = unaCantidad
}

method cantidadRaquetas(){
    return cantidadRaquetas
}

method cantidadRaquetas(unaCantidad){
    cantidadRaquetas = unaCantidad
}

method presupuesto(){
    return self.presupuestoPropio() + disciplina.presupuestoBase(self)
}

method presupuestoPropio(){
    return costoPorRaqueta * self.cantidadRaquetas()
}

}

object pareto {
    var nombre = "Paula Pareto"
    var edad
    var disciplina = judo
    var sparringsLesionados = 1
    var altura = 160
    const costoPorAltura = 50

    method sparringsLesionados() {
        return sparringsLesionados
    }

    method sparringsLesionados(_sparringsLesionados) {
        sparringsLesionados = _sparringsLesionados
    }

    method presupuesto(){
        return self.presupuestoPropio() + disciplina.presupuestoBase(self)
    }

    method presupuestoPropio(){
        return costoPorAltura * altura
    }
}

```

```
}  
}
```

Archivo comite.wlk

```
import deportistas.*  
  
object comite {  
    var fondos = 60000  
    var atleta1 = delpo  
    var atleta2 = pareto  
  
    method fondos() {  
        return fondos  
    }  
  
    method fondos(_fondos){  
        fondos = _fondos  
    }  
  
    method presupuestoTotalDeAtletas(){  
        return atleta1.presupuesto() + atleta2.presupuesto()  
    }  
  
    method estaEnAlertaAmarilla(){  
        return self.presupuestoTotalDeAtletas() > fondos * 0.4 and  
!self.estaEnAlertaNaranja()  
    }  
  
    method estaEnAlertaNaranja(){  
        return self.presupuestoTotalDeAtletas() > fondos * 0.7  
    }  
}
```

Archivo testPresupuesto.wtest

```
import deportistas.*  
import comite.*  
  
test "si le pido un presupuesto a un tenista valido, me lo devuelve el presupuesto esperado"  
{  
    //Precodincion
```

```

        assert.that(delpo.cantidadRaquetas() > 0)

        //Reglas
        assert.equals(126000,delpo.presupuesto())
    }

    test "si le pido un presupuesto a un judoca valido, me lo devuelve el presupuesto esperado"
    {
        //Precodincion
        assert.that(pareto.sparringsLesionados() > 0)

        //Reglas
        assert.equals(26000,pareto.presupuesto())
    }

    test "Si el comité tiene un 40% de fondos superiores al presupuesto de todos los deportistas
    no está en alerta amarilla" {
        comite.fondos(2000000)
        assert.notThat(comite.estaEnAlertaAmarilla())
    }

    test "Si el comité tiene un 40% de fondos inferiores al presupuesto de todos los deportistas
    está en alerta amarilla" {
        comite.fondos(300000)
        assert.that(comite.estaEnAlertaAmarilla())
    }

    test "Si el comité tiene un 70% de fondos superiores al presupuesto de todos los deportistas
    no está en alerta naranja" {
        comite.fondos(2000000)
        assert.notThat(comite.estaEnAlertaNaranja())
    }

    test "Si el comité tiene un 70% de fondos inferiores al presupuesto de todos los deportistas
    está en alerta naranja" {
        comite.fondos(200000)
        assert.that(comite.estaEnAlertaNaranja())
    }

```