

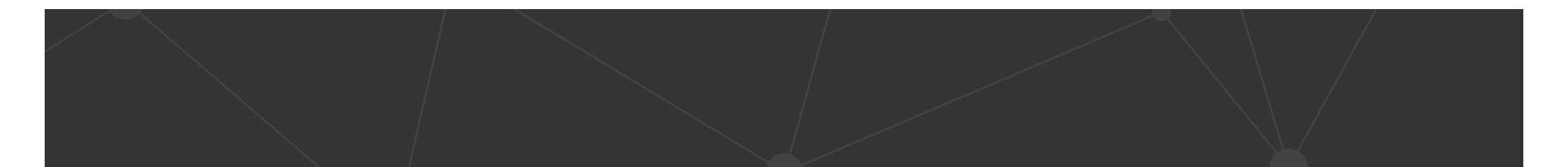


Exercise

Alexandre Bergel

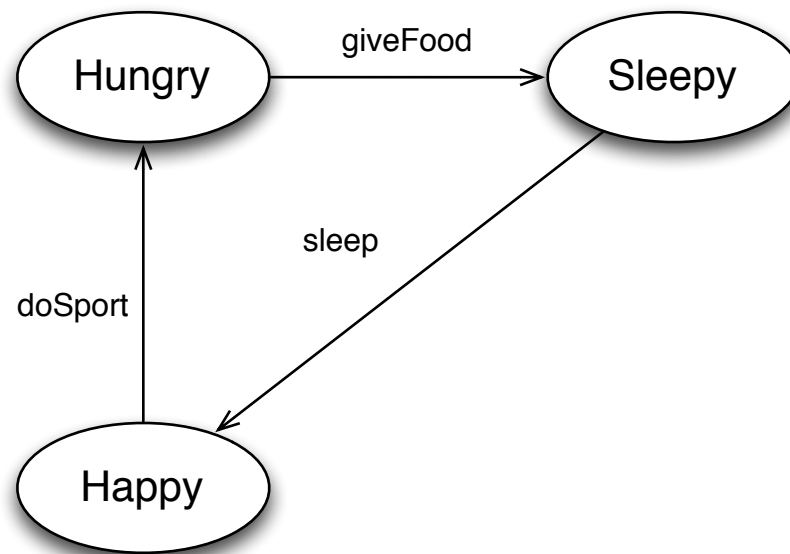
<http://bergel.eu>

18-10-2021



Tamagotchi

Un Tamagotchi es un pequeño animal virtual. Como todo los seres vivos, un tamagotchi tiene necesidades. A continuación se muestra un pequeño diagrama de estado:



Tamagotchi

Al principio un Tamagotchi esta feliz (happy). Si el hace deporte, tiene hambre (hungry). Al darle comida, le da sueño y se pone a dormir (sleepy). Después haber dormido, esta de nuevo feliz.

Exercise

Parte A - Implementa el Tamagotchi.

Parte B - Para poder hacer “tests”, queremos tener una forma de saber en que estado esta el tamagotchi.

Por ejemplo, tenemos `new Tamagotchi().isHappy()` que retorna `true`.

Tenemos `Tamagotchi t = new Tamagotchi(); t.doSport(); t.isHungry()` retorna `true` y `t.isHappy()` retorna `false`. Proponga una implementación para los métodos `isHappy()`, `isHungry()` y `isSleepy()`.

A solution

```
public class Tamagotchi {  
    private State state;  
  
    public Tamagotchi() { this.setState(new Happy()); }  
  
    public boolean isHappy() { return state.isHappy(); }  
    public boolean isHungry() { return state.isHungry(); }  
    public void doSport() { state.doSport(); }  
    public void giveFood() { state.giveFood(); }  
    public void sleep() { state.sleep(); }  
    public boolean isSleepy() { return state.isSleepy(); }  
  
    public void setState(State s) {  
        state = s;  
        s.setTamagotchi(this);  
    }  
}
```

A solution

```
public abstract class State {  
    protected Tamagotchi tama;  
  
    public boolean isHappy() { return false; }  
    public boolean isHungry() { return false; }  
    public boolean isSleepy() { return false; }  
  
    public void error() { throw new AssertionError("Wrong state!"); }  
    public void doSport() { this.error(); }  
    public void giveFood() { this.error(); }  
    public void sleep() { this.error(); }  
  
    public void setState(State aState) { tama.setState(aState); }  
  
    public void setTamagotchi(Tamagotchi tamagotchi) { tama =  
tamagotchi; }  
}
```

A solution

```
public class Happy extends State {  
    public boolean isHappy() { return true; }  
    public void doSport() { this.setState(new Hungry()); }  
}
```

```
public class Hungry extends State {  
    public boolean isHungry() { return true; }  
    public void giveFood() { this.setState(new Sleepy()); }  
}
```

```
public class Sleepy extends State {  
    public boolean isSleepy() { return false; }  
    public void sleep() { this.setState(new Happy()); }  
}
```

A solution

```
public class TamagotchiTest {  
    private Tamagotchi tamagotchi;  
  
    @BeforeEach  
    public void setUp() { tamagotchi = new Tamagotchi(); }  
  
    @Test  
    public void testHappy() {  
        assertTrue(tamagotchi.isHappy());  
    }  
  
    @Test  
    public void testDoSport() {  
        tamagotchi.doSport();  
        assertFalse(tamagotchi.isHappy());  
        assertTrue(tamagotchi.isHungry());  
    }  
  
    @Test  
    public void testWrongState() {  
        AssertionError e = assertThrows(AssertionError.class, () ->  
tamagotchi.giveFood());  
        assertEquals("Wrong state!", e.getMessage());  
    }  
}
```

License



Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)

You are free to:

- Share: copy and redistribute the material in any medium or format
- Adapt: remix, transform, and build upon the material for any purpose, even commercially

The licensor cannot revoke these freedoms as long as you follow the license terms



Attribution: you must give appropriate credit



ShareAlike: if you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original

Complete license: <https://creativecommons.org/licenses/by-sa/4.0/>



dcc

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

www.dcc.uchile.cl

f @ in / DCCUCHILE