PA1458 Example II

Mikael Svahnberg

March 1, 2021

Contents

1 System Description

Desktop Ponies is an application that allows MLP ponies to run around on the screen and execute different behaviours.

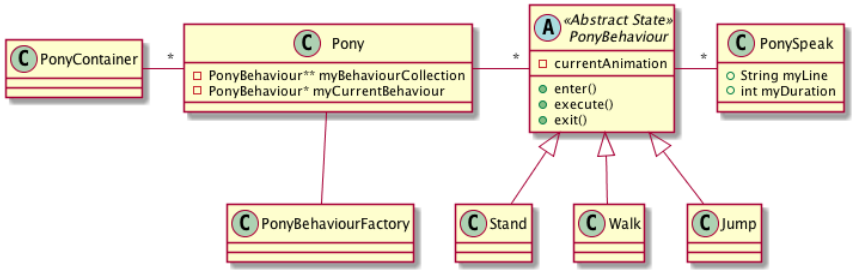
Each pony randomly changes behaviour to do something else. Each behaviour is a **state**.

Link: `https://github.com/mickesv/JSPonies`

2 Class Diagram

```
PonyContainer - "*" Pony

Pony - "*" PonyBehaviour

abstract class PonyBehaviour <<Abstract State>> {
```

```
 +enter()
 +execute()
 +exit()
 -currentAnimation
}

Pony : -PonyBehaviour** myBehaviourCollection
Pony : -PonyBehaviour* myCurrentBehaviour

Pony -- PonyBehaviourFactory

PonyBehaviour <|-- Stand
PonyBehaviour <|-- Walk
PonyBehaviour <|-- Jump

PonyBehaviour - "*" PonySpeak

PonySpeak : +String myLine
PonySpeak : +int myDuration
```



3 Description of Classes

Pony is in the State pattern. It owns a collection of `PonyBehaviour` and has a `currentPonyBehaviour` which is the currently active state.

PonyBehaviour is the . This class provides an interface that all concrete states have to implement.

{Stand, Walk, Jump} are the . Each implement a state and its behaviour in the `enter()` `exit()` and `execute()` methods.

PonySpeak is responsible for one single line of speech that a PonyBehaviour can say.

PonyContainer contains all ponies.

PonyBehaviourFactory Given a behaviour name, it creates an object based on one of the concrete implementations of PonyBehaviour.

4 Pseudocode

```
Pony::create() {
 String** behaviourNames = PonyInitFile::getBehaviours(myPonyName);

 behaviourNames.forEach( (n) => {
   PonyBehaviour* pb = PonyBehaviourFactory::createBehaviour(n);
   myBehaviourCollection.append(pb);
 });

 myCurrentBehaviour = myBehaviourCollection[0];
 myCurrentBehaviour->enter();
}


Pony::setBehaviour(String newBehaviourName) {
  PonyBehaviour pb = myBehaviourCollection.find(newBehaviourName);
  if (pb) {
    myCurrentBehaviour->exit();
    myCurrentBehaviour = pb;
    myCurrentBehaviour->enter();
  }
}
```

5 Discussion of GRASP patterns

- The context class is a **Creator** of the different states
- The context class is an **Information Expert** about which states exist, and which is the current state.
- The abstract state is an **Information Expert** about the interface that each state must provide

- The concrete states are **Information Experts** on what it means to be in that particular state

6 Usage of GRASP patterns

- PonyBehaviourFactory is a **Creator** of PonyBehaviours
- PonyBehaviour, as is an **information expert** on the interface to the states