HomeExam Example II

Mikael Svahnberg*

2020-05-14

Contents

1 Svar på fråga: Unit Testing

- testsvit
  - setup
  - test
    - setup
    - test
    - teardown
  - teardown
- Test system events

  selectObject() input a string with the name of a GameObject. returns an array with strings corresponding to InteractionTypes

```
TEST {
 // setup
 Game g =  new Game();
 g.addObject(new GameObject("testObject", ["test", "look"]));:

 // Test
 String[] result = g.selectObject("testObject");
```

*Mikael.Svahnberg@bth.se

```
 assert_eq(result, ["test", "look"]);
 assert(notEmpty(result));

 // Teardown
 g.removeObject("testObject");
 delete [] g;
}

TEST {
 // setup
 Game g =  new Game();
 g.addObject(new GameObject("testObject", ["test", "look"]));:

 // Test
 String [] result = g.selectObject("ThisObjectMustNotExist");
 assert(gotError(result));
 assert(isEmpty(result));

 // Teardown
}
```

- selectInteractionType()
- selectInteractionOptions()
- startInteraction()

2 Exempeltenta: State Pattern

2.1 Uppgifter

Samma som innan, fast med **State Pattern**

1. Uppgift 1
   (a) Hitta på ett mindre system där du kan använda designmönstret **State pattern**. Beskriv systemet och hur du tänker använda State pattern för att lösa en del av systemet.
   (b) Gör ett klassdiagram med klasser, attribut och metoder för den del av systemet där ditt State pattern används.
   (c) Beskriv hur du använt State pattern och vilka klasser och metoder som är inblandade. Vilka roller har de? Vilka ansvarsområden har de?
   (d) Skriv pseudokod (eller kod i Java eller C++) för de metoder där du skapar objekt enligt ditt State pattern och de metoder där du använder dig av dessa objekt.
2. Uppgift 2
   (a) På vilket sätt är State Pattern relaterat till GRASP-mönstren **Information Expert** och **Controller**? Beskriv kortfattat.

(b) Använd ditt system från uppgift 1 och beskriv hur Information Expert och Controller används för att fördela ansvar mellan klasserna i ditt system.

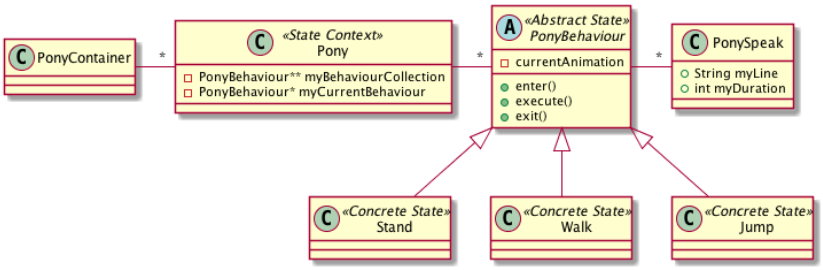
2.2 Systembeskrivning

Desktop Ponies är en applikation som låter MLP ponies springa runt på skärmen och utföra olika saker.

Varje Ponny byter vid slumvisa tillfällen beteenden och gör någonting annat. Varje beteende är ett **tillstånd**, ett **state**.

2.3 Klassdiagram

```
class Pony <<State Context>>
PonyContainer - "*" Pony

abstract class PonyBehaviour <<Abstract State>> {
 +enter()
 +execute()
 +exit()
 -currentAnimation
}

Pony - "*" PonyBehaviour

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

class Stand <<Concrete State>>
class Walk <<Concrete State>>
class Jump <<Concrete State>>

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

PonyBehaviour - "*" PonySpeak

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



2.4 Beskrivning av klasser

Pony är «context» i State pattern. Den äger en samling PonyBehaviours, och har ett currentBehaviour som är dert nu gällande statet.

PonyBehaviour är «Abstract State». Den erbjuder ett gränssnitt med de metoder som varje tillstånd skall implementera.

{Stand, Walk, Jump} är «Concrete State» i state pattern. Var och en implementerar ett tillstånd, och har det beteendet i sina enter, execute och exit-metoder.

PonySpeak en replik som en Ponny kan säga i vissa tillstånd.

PonyContainer innehåller alla Ponnies.

2.5 Pseudokod

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

 BehaviourNames.forEach( (n) => {
   PonyBehaviour*  pb = new PonyBehaviour(n);
   myPonyBehaviourCollection.append(pb);
});
}

Pony::setBehaviour(string newBehaviour) {

  PonyBehaviour* newBehaviour = myBehaviourCollection->find(newBehaviour);

  if (newBehaviour) {
   myCurrentBehaviour->exit();
   myCurrentBehaviour = newBehaviour;
   myCurrentBehaviour->enter();
  }
}
```

2.6 GRASP patterns

PonyContainer är Information Expert över vilka Ponies som finns.

Pony är Information Expert över vilka tillstånd en viss ponny har och vilket tillstånd som gäller just nu.

Pony är Controller för vad som skall göras i ett visst ögonblick.

PonyBehaviour är information expert över vilket gränssnitt varje beteende skall ha

{Stand, Walk, Jump} är information experts på vad det innebär att vara i just det tillståndet.

PonySpeak är Information Expert på en specifik sak som en specifik ponny kan säga i ett specifikt tillstånd.