Summary

Project Name

Patterns

Project Repository

https://github.com/Lukeekul/wahlzeit

Project CI

https://travis-ci.org/Lukeekul/wahlzeit

Current Tag

adap-hw10 on master

Diff to last tag

https://github.com/Lukeekul/wahlzeit/compare/adap-hw09-3...Lukeekul:adap-hw10 https://github.com/Lukeekul/wahlzeit/compare/adap-hw09...Lukeekul:adap-hw10 (containing corrections from last homework)

Overview

- Adding new Annotation *PatternInstance*
- Adding Annotation to five Design Pattern Instances

Details

Implementation of Annotation

- The Annotation *PatternInstance* is impelemented as suggested in lecture and in slides in package *org.wahlzeit.annotaion*
- An additional Annotation *PatternInstances* is implemented to allow for multiple *PatternInstance* Annotations at a single class.

Annotation of Design Pattern Instances

- Singleton Pattern: **PhotoFactory**, **PhotoManager**: Both PhotoFactory and PhotoMananer should be unique, since PhotoManager is generating unique IDs for Photo and PhotoFactory is creating Photo Objects that uses these IDs.

- Factory Pattern: **PhotoFactory** along with Photo, **PatternPhotoFactory** along with PatternPhoto: For both factories, PhotoFactory and the specialized PatternPhotoFactory, the creation process is hidden in the factories. To get a Photo/PatternPhoto object, the method **createPhoto** can be called without the need to call a Constructor of a Photo/PatternPhoto leaf class.
- Flyweight Pattern: **CoordinateFactory** along with CartesianCoordinate and SphericCoordinate: In order to implement Coordiante as Value Object, the Flyweight Pattern Approach was chosen. CartesianCoordinate and SphericCoordinate both provide static methods that return instances of their own type. The CoordinateFactory manages the object creation for both types of Coordinate to ensure new objects are only created if a Coordinate with the requested parameters does not exsisted yet. Otherwise the existing object is returned.

Corrections for last homework (adap-hw09)

- A new Class **CoordinateFactory** is added, that takes care of HashMap handling of the instances for each typeof Coordinates. The Constructor of each Coordinate leaf class are private, their member variables final. Through the public method **getCoordinate** an instance is returned. By this approach, a Coordinate instance can still be created without the factory, if not desired as Value Object. The Factory will check first if an instance with the requested parameters and type already exists and will create will call the **getCoordinate** method only if that is not the case. Otherwise the already existing object from the HashMap will be returned.