Composite

By Alexander Tilkin

Motivation

- Objects use other objects' fields/methods through inheritance and composition
- Composition lets us make compound objects
  - E.g., mathematical expression composed of simple expressions; or
  - A shape group made of several different shapes
- Composite design pattern is used to treat both single (scalar) and composite objects uniformity
  - I.e, Foo and List<Foo> have common APIs

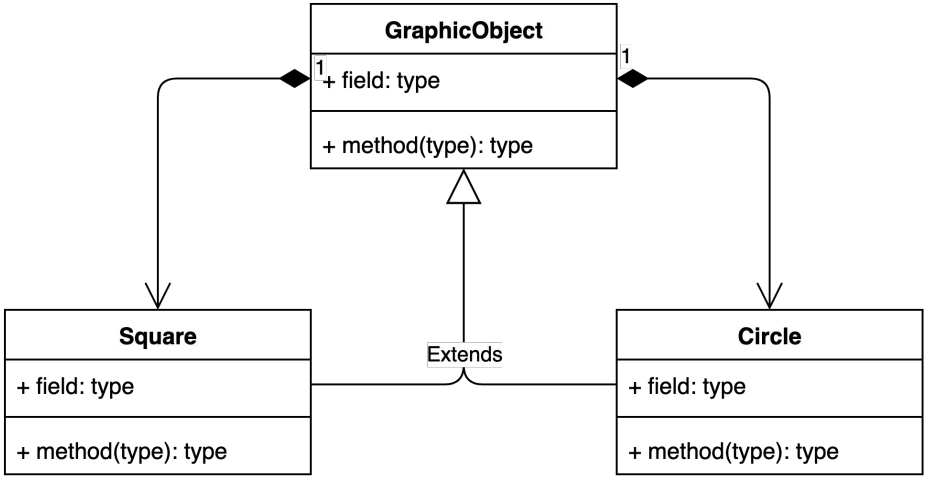
Composite

A mechanism for treating individual (scalar) objects and compositions of objects in a uniform manner.

units.unit_08_composite.GraphicShapesExample.java

Graphic Shapes Example

- GraphicObject contains a collection of GraphicObject items
- Square and Circle extends GraphicObject
- This allows GraphicObject access both Circle and Sqaure classes
- Using this structure we can iterate over a hierarchy of objects



units.unit_08_composite.CompositeExercise.java

Exercise

- Consider the code presented in the link above
- The MyList.sum() method adds up all the values in a list of ValueContainer elements it gets passed. We can have a single value or a set of values
- Complete the implementation of the interfaces so that sum() begins to work correctly

Summary

- Objects can use other objects via inheritance/composition
- Some composed and singular objects need similar/identical behaviors
- Composite design pattern lets us treat both types of objects uniformity
- Java supports container iteration with the Iterable<T> interface
- A single object can masquerade as a collection be returning a single-element collection containing only *this*