Design Patterns

Composite Pattern

ebru@hacettepe.edu.tr
ebruakcapinarsezer@gmail.com
http://yunus.hacettepe.edu.tr/~ebru/
@ebru176
Kasım 2017

Composite Pattern

- a Composite is an object (e.g. a shape) designed as a composition of one-or-more similar objects (other kinds of shapes/geometries),
- all exhibiting similar functionality
- this is known as a "has-a" relationship between objects
- the key concept is that you can manipulate a single instance of the object just as you would a group of them.

Composite (from Gof)

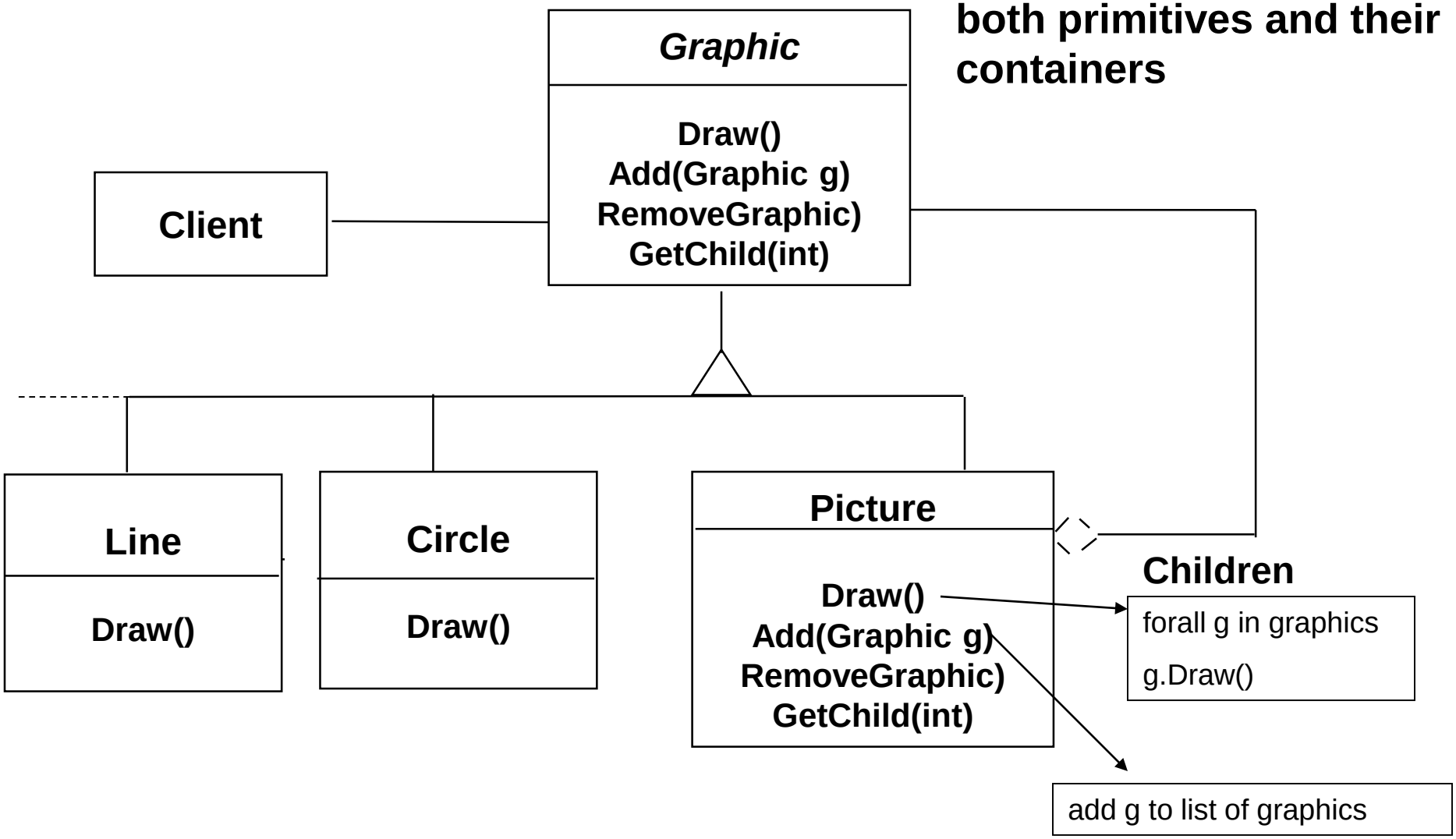
Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly

Composite Problem

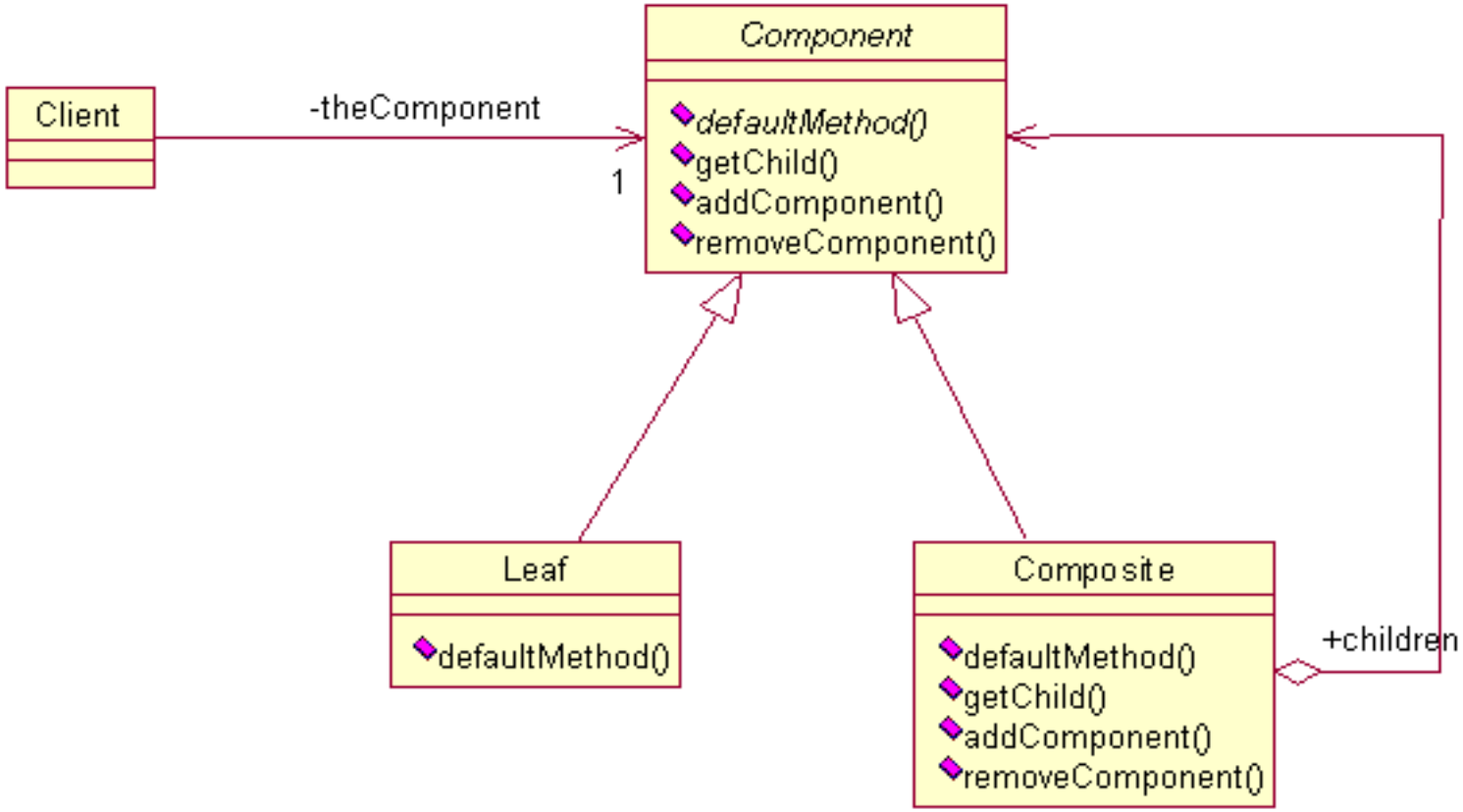
- In graphics applications, the user can group components to form larger components
- A simple implementation: define classes for graphical primitives such as Text and Lines plus other classes that act as containers for these primitives.
- Code that uses these classes must treat primitives and container objects differently, even if most of the time the user treats them identically.

Composite

The key to the composite pattern is an abstract class that represents both primitives and their containers



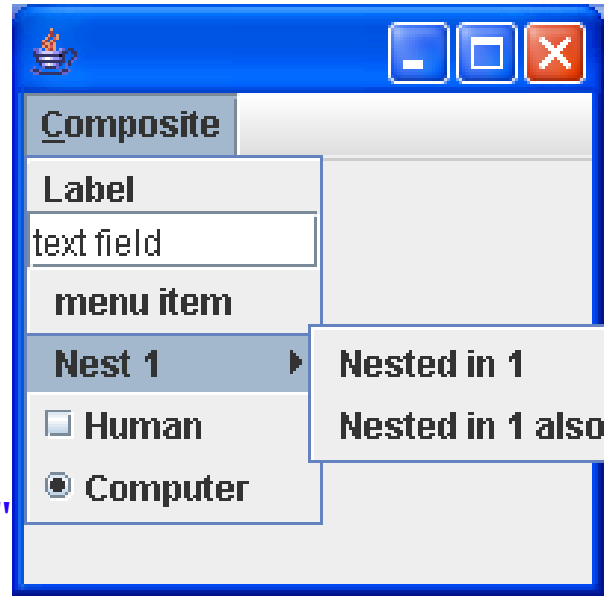
Structure


Client
-theComponent
1
Component
defaultMethod()
getChild()
addComponent()
removeComponent()
Leaf
defaultMethod()
Composite
defaultMethod()
getChild()
addComponent()
removeComponent()
+children

Details

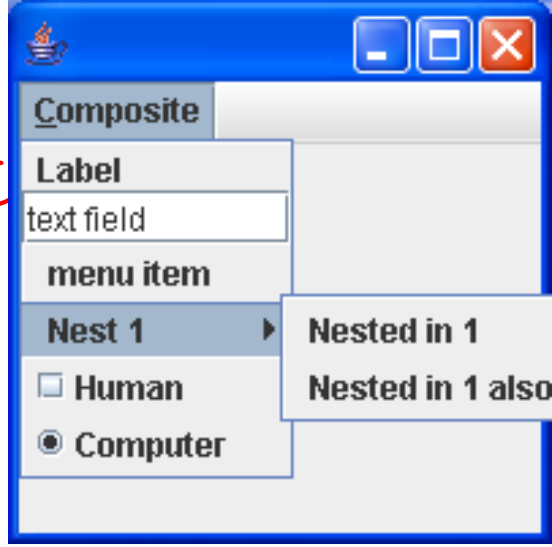
- Component
  - declares the interface for object composition
  - implements default behaviour
  - declares an interface for accessing and managing the child components
- Leaf (Feuille)
  - represents leaf objects in the composition
- Composite
  - defines behaviour for components having children
  - stores child components
  - implements child-related operations to the Component interface
- Client
  - manipulates objects in the composition through the Component interface

Swing based Composite Ex




```
JMenuItem menu = new JMenu("Composite");
menu.setMnemonic('C');//Open with alt-C
// Create two leafs
JLabel label = new JLabel("Label");
JTextField textF = new JTextField("text field"
menu.add(label);
menu.add(textF);
// Add a Composite
JMenuItem menuItem = new JMenuItem("menu item");
menu.add(menuItem);
// Add two Composites to a Composite
JMenuItem jmi1Nest = new JMenu("Nest 1");
menu.add(jmi1Nest);
JMenuItem jmiNested1 = new JMenuItem("Nested in 1");
jmi1Nest.add(jmiNested1);
JMenuItem jmiNested2 = new JMenuItem("Nested in 1 also");
jmi1Nest.add(jmiNested2);
```

JMenuItemDemoComposit




```
// Add two more Composites
JMenuItem checkBox
     = new JCheckBoxMenuItem("Human", false);
JMenuItem radioButton
     = new JRadioButtonMenuItem("Computer", true);
menu.add(checkBox);
menu.add(radioButton);
// Add two more Composites
JMenuBar menuBar = new JMenuBar();
setJMenuBar(menuBar);
menuBar.add(menu);
```

Run `JMenuItemDemoComposite.java`

See code demo page