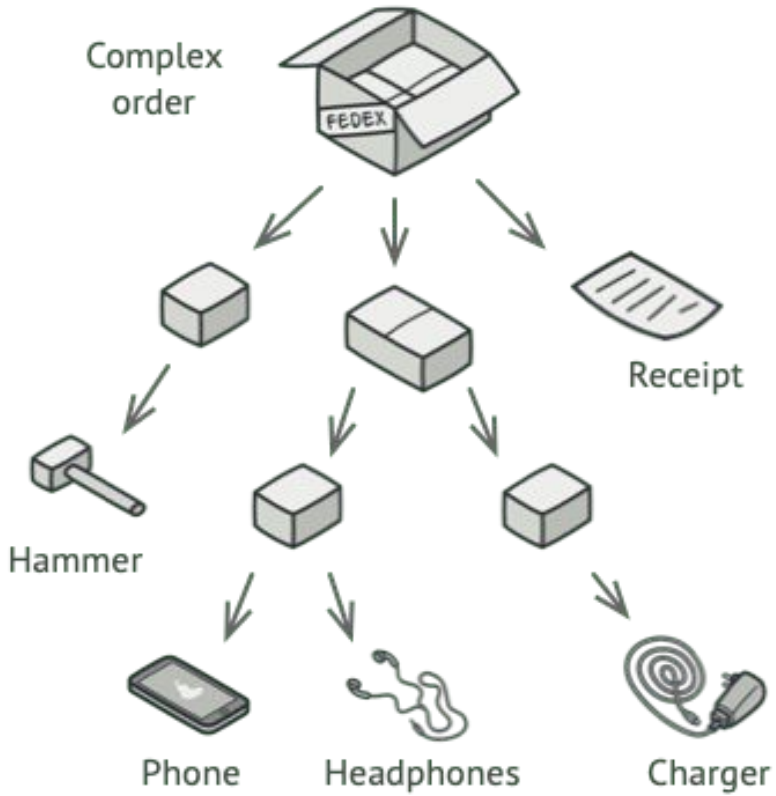
Composite and Command Pattern

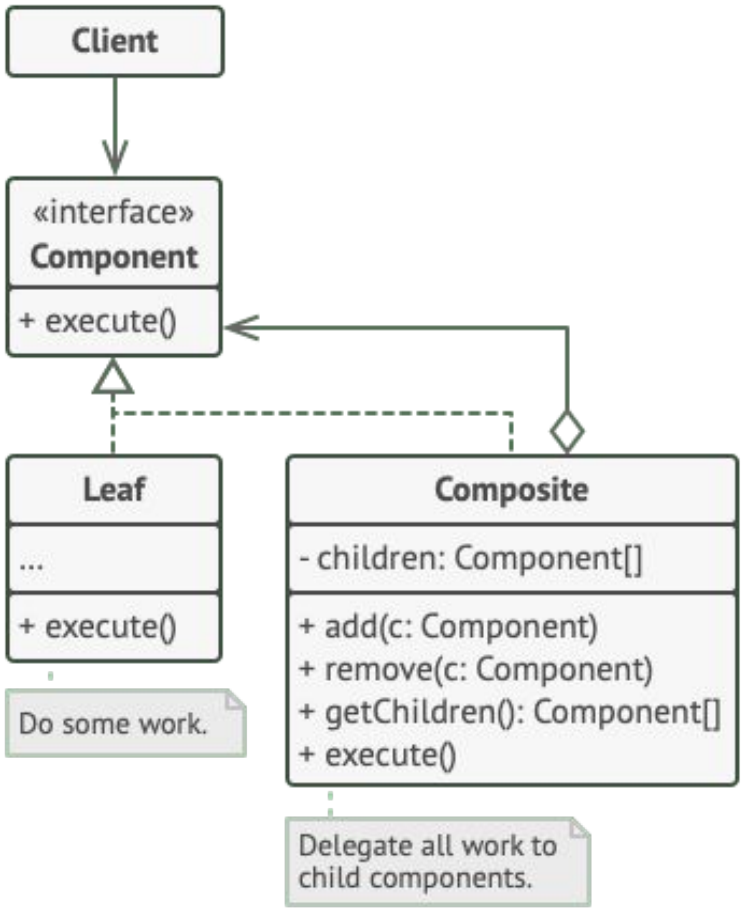
Composite pattern

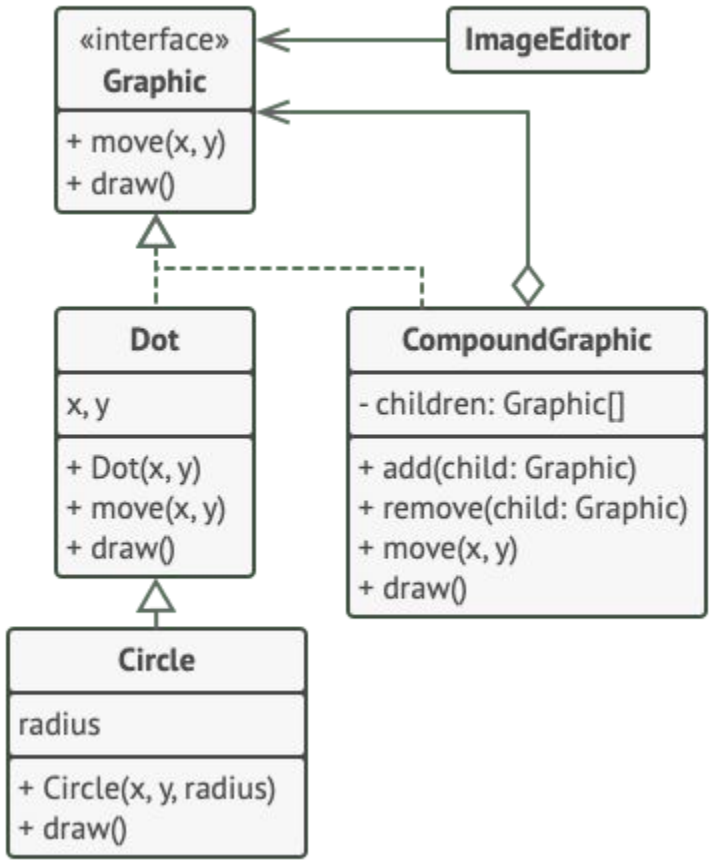
- Composite pattern is used where we need to treat a group of objects in similar way as a single object.
- Composite pattern composes objects in term of a tree structure to represent part as well as whole hierarchy.
- This type of design pattern comes under structural pattern as this pattern creates a tree structure of group of objects.
- This pattern creates a class that contains group of its own objects. This class provides ways to modify its group of same objects.

Composite pattern



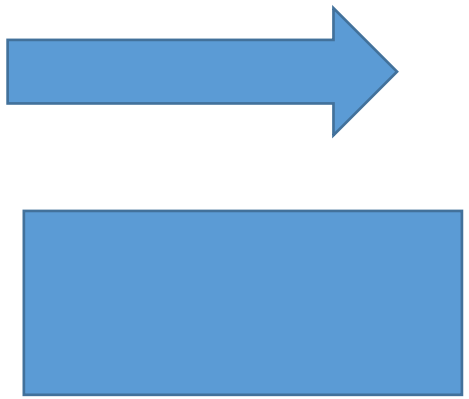
Composite pattern





Composite Pattern

- New line() new square() composite(line, square)



Command Pattern

- Command pattern is a data driven design pattern and falls under behavioral pattern category.
- A request is wrapped under an object as command and passed to invoker object. Invoker object looks for the appropriate object which can handle this command and passes the command to the corresponding object which executes the command.

Command Pattern

Stock.java

```
public class Stock {

   private String name = "ABC";
   private int quantity = 10;

   public void buy(){
      System.out.println("Stock [ Name: "+name+",
         Quantity: " + quantity +" ] bought");
   }
   public void sell(){
      System.out.println("Stock [ Name: "+name+",
         Quantity: " + quantity +" ] sold");
   }
}
```

Command Pattern

Order.java

```
public interface Order {
   void execute();
}
```

BuyStock.java

```
public class BuyStock implements Order {
   private Stock abcStock;

   public BuyStock(Stock abcStock){
      this.abcStock = abcStock;
   }

   public void execute() {
      abcStock.buy();
   }
}
```

SellStock.java

```
public class SellStock implements Order {
   private Stock abcStock;

   public SellStock(Stock abcStock){
      this.abcStock = abcStock;
   }

   public void execute() {
      abcStock.sell();
   }
}
```

Command Pattern

Broker.java

```
import java.util.ArrayList;
import java.util.List;

   public class Broker {
   private List<Order> orderList = new ArrayList<Order>();

   public void takeOrder(Order order){
      orderList.add(order);
   }

   public void placeOrders(){

      for (Order order : orderList) {
         order.execute();
      }
      orderList.clear();
   }
}
```

Command Pattern

CommandPatternDemo.java

```
public class CommandPatternDemo {
   public static void main(String[] args) {
      Stock abcStock = new Stock();

      BuyStock buyStockOrder = new BuyStock(abcStock);
      SellStock sellStockOrder = new SellStock(abcStock);

      Broker broker = new Broker();
      broker.takeOrder(buyStockOrder);
      broker.takeOrder(sellStockOrder);

      broker.placeOrders();
   }
}
```

UML Command Pattern