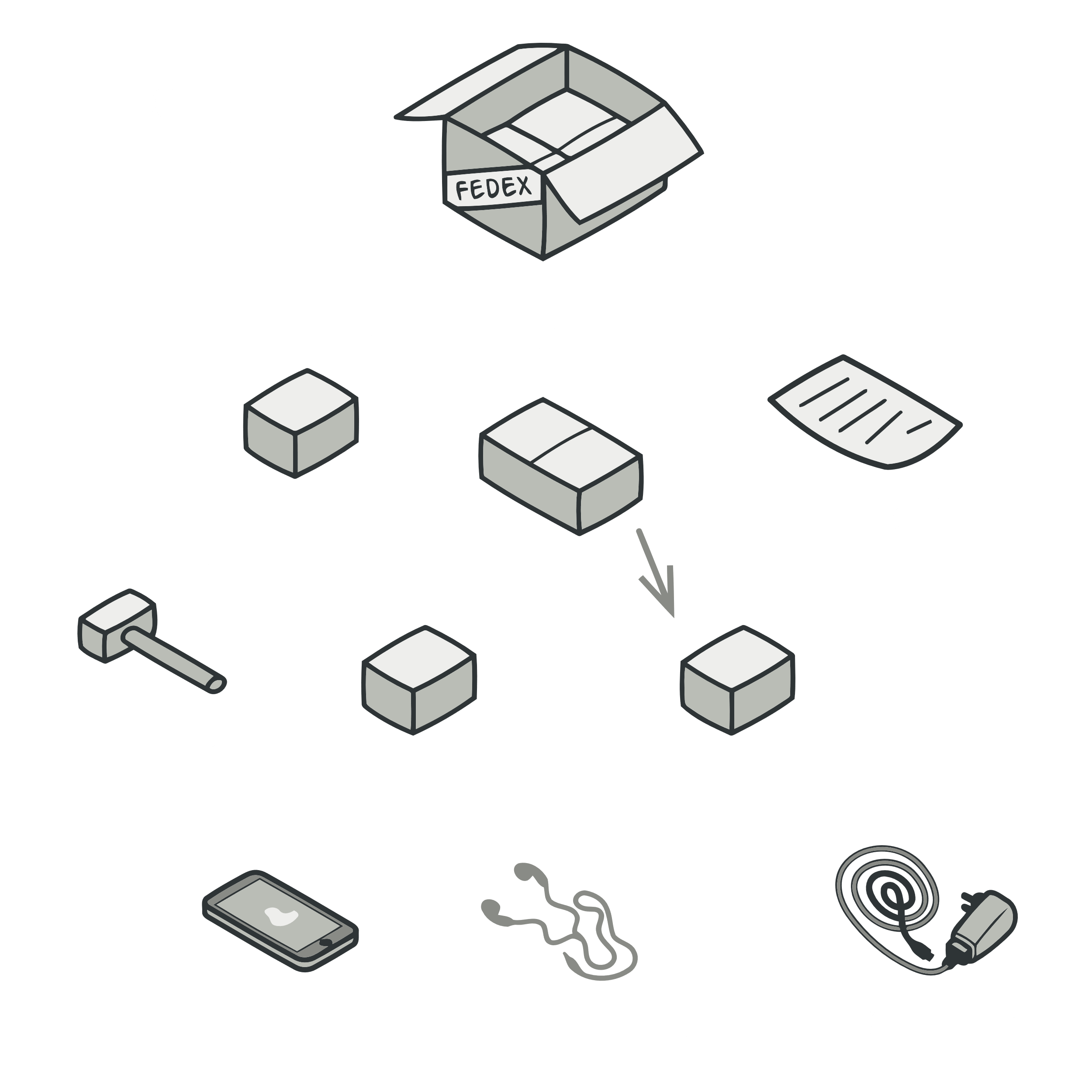
Composite Pattern

The **composite pattern** is used when we need to treat a group of objects in the same way that we treat one of the individual objects. This is done by composing the objects as a tree structure. It is a **structural pattern**. We create a class that contains groups of its own objects.



public class Employee {  
 private String name;  
 private String dept;  
 private int salary;  
 private *List*<Employee> subordinates;  
  
 public Employee(String name,String dept, int sal) {  
 this.name = name;  
 this.dept = dept;  
 this.salary = sal;  
 subordinates = new ArrayList<>();  
 }

public void add(Employee e) {  
 subordinates.add(e);  
 }  
  
 public void remove(Employee e) {  
 subordinates.remove(e);  
 }  
  
 public *List*<Employee> getSubordinates() {  
 return subordinates;  
 }  
  
 public String toString() {  
 return ("Employee :[ Name : " + name + ", dept : " + dept + ", salary :" + salary+" ]");  
 }  
}  
  
public class CompositePatternDemo {  
 public static void main(String[] args) {  
  
 Employee CEO = new Employee("John","CEO", 30000);  
  
 Employee headSales = new Employee("Robert","Head Sales", 20000);  
  
 Employee headMarketing = new Employee("Michel","Head Marketing", 20000);  
  
 Employee clerk1 = new Employee("Laura","Marketing", 10000);  
 Employee clerk2 = new Employee("Bob","Marketing", 10000);  
  
 Employee salesExecutive1 = new Employee("Richard","Sales", 10000);  
 Employee salesExecutive2 = new Employee("Rob","Sales", 10000);  
  
 CEO.add(headSales);  
 CEO.add(headMarketing);  
  
 headSales.add(salesExecutive1);  
 headSales.add(salesExecutive2);  
  
 headMarketing.add(clerk1);  
 headMarketing.add(clerk2);  
  
 System.*out*.println(CEO);

for (Employee headEmployee : CEO.getSubordinates()) {  
 System.*out*.println(headEmployee);  
  
 for (Employee employee : headEmployee.getSubordinates()) {  
 System.*out*.println(employee);  
 }  
 }  
 }  
}

JAVA

