ANADOLU ÜNİVERSİTESİ

Design Pattern Project 3

Hana Hadžo Mulalić

987 156 287 27

prof. Dr. Öğr. ÜyesiAlper BİLGE

Res.Asst. Gökhan ÇIPLAK

Solve a software development problem using one (or possibly more) of the design patterns that we cover in class. Your problem should be different from your previous projects.

To-Do

Statement of Work: Write your problem definition in detail using 100-200 words.

Design Pattern(s): Select design pattern(s) that are useful in solving the problem you defined. Explain why and how you employ such design pattern(s) in your Project.

UML: Draw a detailed class diagram of your proposed solution using some UML Diagram Drawing Tool.

Implementation: Implement your proposed solution based on the UML Class Diagram using the base Maven Project that you are supplied. Your project should not expect any input from user.

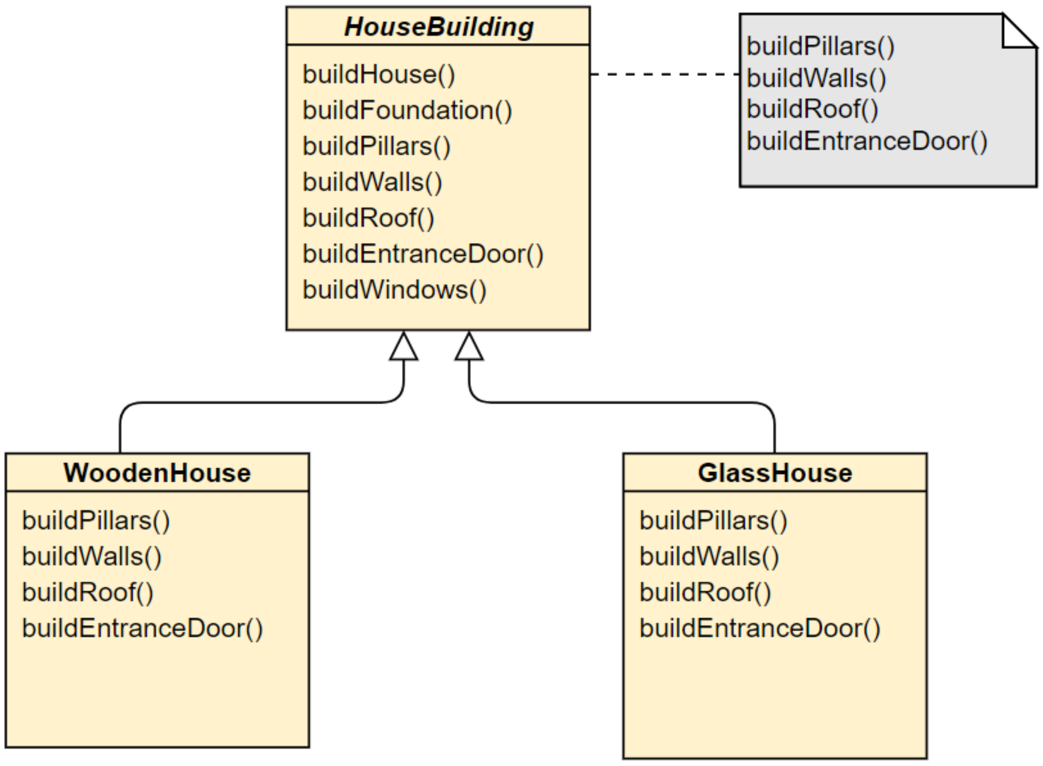
Statement of Work

It is needed to implement java program HouseBuilding which provide an algorithm for building of the house. Steps for house building, which are methods in the code, are foundation building, pillars building, walls building, roof building entrance door building and windows building. What is important, is to care about not to change the steps order (it is not logical to build windows before the house walls). More, it is needed to add two subclasses WoodenHouse and GlassHouse which extend Housebuilding superclass. Both subclasses should follow the same algorithm-methods from the superclass. Furthermore, foundation building step is the same for both kinds of houses (built from the same mixture). Also, for both buildings, the windows building step is the same (glass windows). Therefore, those two methods should be inherited in the subclasses WoodenHouse and GlassHouse. Other methods need to be overridden in the subclasses.

Design Pattern

For such defined problem, the most effective way to solve it is using **Template Method Pattern**. Each step of house building implements a separate method. There are methods to build foundation, build pillars, build walls, build roof, build entrance door and build windows. Some of them are common for both, WoodeHouse and GlassHouse, and others are specialized to WoodenHouse or to GlassHouse. We should abstract the commonality into a base, superclass since WoodeHouse and GlassHouse are very similar. buildFoundation () and buildWindows() should be abstracted into the base class, since they are the common in both superclasses. Other methods are not abstracted, but are the same instead they apply to diferent kinds of houses. Therefore, we can abstract buildHouse() too. For example walls building is the step of algorithm for the both houses, just the different materials (wood / glass) are used for building. So, we need to make a new, common method – buildWalls(). The similar is for others: buildPillars(), buildRoof()… Now, the same buildHouse() will be used to build both wooden and glass house. It should be defined as final since we do not want the subclasess to override it. In problem definition, it is written the steps should not be changed. Since the WoodenHouse and GlassHouse handle some mentioned differently, we need to declare them as abstract. The common methods buildFoundation () and buildWindows() should be declared into HouseBuilding class. Implementation of buildHouse() should be also in this HouseBuilding abstract class. It is the template method, which is “template” for algorithm. As we see, the Template Method defines the steps of an algorithm and allows subclasses to provide the implementation for more steps.

UML Diagram



Implementation

HouseBuilding

```
package housebuilding;

/**
 *
 * @author Hana
 */
public abstract class HouseBuilding {
    final void buildHouse(){
        buildFoundation();
        buildPillars();
        buildWalls();
        buildRoof();
        buildEntranceDoor();
        buildWindows();

    }

abstract void buildPillars();
abstract void buildWalls();
abstract void buildRoof();
abstract void buildEntranceDoor();


void buildFoundation(){
System.out.println("Building the house foundation from mixture of water, gravel, sand, cement and 
nets.");
}

void buildWindows() {
System.out.println("Installation of the glass windows.\n \n");
}
}
```

WoodenHouse

```
package housebuilding;

/**
 *
 * @author Hana
 */
public class WoodenHouse extends HouseBuilding {
    public void buildPillars(){
        System.out.println("Placing of the wooden pillars in order to build house framework.");
    }
    public void buildWalls() {
        System.out.println("Building wooden walls.");
    }
    public void buildRoof() {
       System.out.println ("Building wooden roof.");
    }
    public void buildEntranceDoor(){
       System.out.println("Installation of the wooden entrance door.");
    }

}
```

GlassHouse

```
package housebuilding;

/**
 *
 * @author Hana
 */
public class GlassHouse extends HouseBuilding {
     public void buildPillars(){
       System.out.println("Placing of the metal pillars in order to build house framework.");
    }
    public void buildWalls() {
       System.out.println("Building glass walls.");
    }
    public void buildRoof() {
       System.out.println ("Building glass roof.");
    }
    public void buildEntranceDoor(){
       System.out.println("Installation of the glass entrance door.");
    }
}
```

HouseBuildingTest

```
package housebuilding;

/**
 *
 * @author Hana
 */
public class HouseBuildingTest {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        WoodenHouse woodenHouse = new WoodenHouse();
        GlassHouse glassHouse = new GlassHouse();

        System.out.println ("¨¨Building wooden house¨¨¨\n");
        woodenHouse.buildHouse();

        System.out.println ("¨¨Building Glass House¨¨\n");
        glassHouse.buildHouse();

    }

}
```