Abstract Factory Design Pattern

By Joshua Neutrom (#39), Dayu Wang (#59), Chen Wang (#58), Yunlong Liu (#25)

Introduction

- Definition
- Relationship
- Benefits
- Negatives
- Illustration
- UML
- Code Example
- Input/Output of Code
- Questions

Abstract Factory Design Pattern Definition

- Creational Design Pattern
- Creates and knows families of related objects (wrapper)
- Platform independence
- Factory method per product
- Class derived for each platform
- Centralized creation for decentralized instantiation

Relationship to Other Design Patterns

- Similarities
  - Can be a Singleton
  - Implemented by Factories or Prototypes
- Differences
  - Alternative to a Facade
  - Different production style from a Builder

Negatives

- Adds complexity
- Slower than constructors
- Difficult to troubleshoot

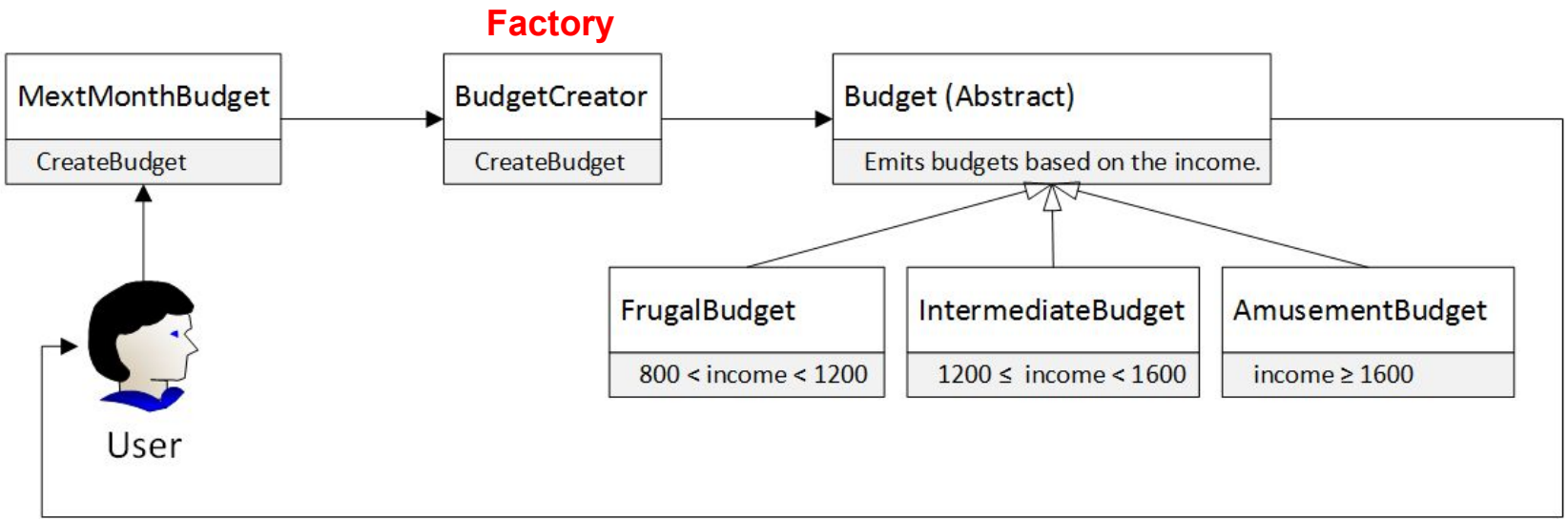
Benefits

- Encapsulation
- Wrapper for various objects
- Platform independence

Illustration

- New User Account Factory
- Method for each service (email, calendar, id number, etc)

Sample Code


Factory
MextMonthBudget
CreateBudget
BudgetCreator
CreateBudget
Budget (Abstract)
Emits budgets based on the income.
FrugalBudget
800 < income < 1200
IntermediateBudget
1200 ≤ income < 1600
AmusementBudget
income ≥ 1600
User

Code

NextMonthBudget.java

```
8  public class NextMonthBudget {
9
10     public static void main(String[] args) {
11
12         BudgetCreator creator = new BudgetCreator();
13         Budget myBudget = null;
14         double income = 0;
15         DateTimeFormatter format = DateTimeFormat.forPattern("yyyy-MM-dd");
16
17         @SuppressWarnings("resource")
18         Scanner userInput = new Scanner(System.in);
19         System.out.println("Please enter your monthly net income\n");
20
21         if (userInput.hasNextLine()) {
22             income = Double.parseDouble(userInput.nextLine());
23             myBudget = creator.CreateBudget(income);
24             System.out.println();
25
26             if (myBudget != null) {
27                 System.out.println("My budget of the next month:");
28                 System.out.println("Income: $" + income);
29                 System.out.println("Start date: " + format.print(myBudget.GetStart()));
30                 System.out.println("End date: " + format.print(myBudget.GetEnd()));
31                 System.out.println("Food budget: $" + Math.round(myBudget.GetFoodBudget()));
32                 System.out.println("Living budget: $" + Math.round(myBudget.GetLivingBudget()));
33                 System.out.println("Entertainment budget: $" + Math.round(myBudget.GetEntertainmentBudget()));
34                 System.out.println("Expected saving: $" + Math.round(myBudget.GetSaving()));
35             }
36             else {
37                 System.out.println("Wrong input!");
38             }
39         }
40     }
41 }
```

Code

NextMonthBudget.java | *BudgetCreator.java

```
1  package team1.activelearning1.budget;
2
3  import org.joda.time.DateTime;
5
6  public class BudgetCreator {
7
8      public Budget CreateBudget(double income) {
9
10         // Get the next month.
11         LocalDate localDate = new LocalDate();
12         DateTime start = localDate.plusMonths(1).withDayOfMonth(1).toDateTimeAtStartOfDay();
13         DateTime end = localDate.plusMonths(1).withDayOfMonth(
14                 localDate.plusMonths(1).dayOfMonth().getMaximumValue())
15                 .toDateTimeAtStartOfDay();
16
17         // Return different budgets based on the income.
18         if (income > 800 && income < 1200) {
19             return new FrugalBudget(income, start, end);
20         }
21         else if (income >= 1200 && income < 1800) {
22             return new IntermediateBudget(income, start, end);
23         }
24         else if (income >= 1800) {
25             return new AmusementBudget(income, start, end);
26         }
27         else {
28             return null;
29         }
30     }
31 }
```

Code

NextMonthBudget.java | BudgetCreator.java | Budget.java

```
4
5  public abstract class Budget {
6
7      // Generic data fields
8      private double monthlyIncome;
9      private DateTime startDate;
10     private DateTime endDate;
11
12     // Data fields that may vary for different budgets
13     private double foodBudget;
14     private double livingBudget;
15     private double entertainmentBudget;
16     private double saving;
17
18     // Generic setters and getters
19     public void SetIncome(double income) {monthlyIncome = income;}
20     public double GetIncome() {return monthlyIncome;}
21     public void SetStart(DateTime start) {startDate = start;}
22     public DateTime GetStart() {return startDate;}
23     public void SetEnd(DateTime end) {endDate = end;}
24     public DateTime GetEnd() {return endDate;}
25     public double GetFoodBudget() {return foodBudget;}
26     public double GetLivingBudget() {return livingBudget;}
27     public double GetEntertainmentBudget() {return entertainmentBudget;}
28
29     public void SetSaving() {
30         saving = monthlyIncome - foodBudget - livingBudget - entertainmentBudget;
31     }
32
33     public double GetSaving() {return saving;}
34     public void SetFoodBudget(double food) {foodBudget = food;}
35     public void SetLivingBudget(double living) {livingBudget = living;}
36     public void SetEntertainmentBudget(double entertainment) {entertainmentBudget = entertainment;}
37 }
```

Code

NextMonthBudget.java | BudgetCreator.java | Budget.java | FrugalBudget.java

```
1  package team1.activelearning1.budget;
2
3  import org.joda.time.DateTime;
4
5  public class FrugalBudget extends Budget {
6
7      public FrugalBudget(double income, DateTime start, DateTime end) {
8
9          SetIncome(income);
10         SetStart(start);
11         SetEnd(end);
12
13         // Calculate food, living, and entertainment budgets.
14         double food = (1.0/2) * income;
15         double living = (1.0/3) * income;
16         double entertainment = (1.0/6) * income;
17
18         SetFoodBudget(food);
19         SetLivingBudget(living);
20         SetEntertainmentBudget(entertainment);
21         SetSaving();
22     }
23 }
24
```

Code

NextMonthBudget.java | BudgetCreator.java | Budget.java | FrugalBudget.java | Intermediate

```java
package team1.activelearning1.budget;

import org.joda.time.DateTime;

public class IntermediateBudget extends Budget {

    public IntermediateBudget(double income, DateTime start, DateTime end) {
        SetIncome(income);
        SetStart(start);
        SetEnd(end);

        // Calculate food, living, and entertainment budgets.
        double food = (1.0/3) * income;
        double living = (1.0/4) * income;
        double entertainment = (1.0/4) * income;

        SetFoodBudget(food);
        SetLivingBudget(living);
        SetEntertainmentBudget(entertainment);
        SetSaving();
    }
}

```

Code

NextMonthBudget.java | BudgetCreator.java | Budget.java | FrugalBudget.java | Intermec

```
1  package team1.activelearning1.budget;
2
3  import org.joda.time.DateTime;
4
5  public class AmusementBudget extends Budget {
6
7      public AmusementBudget(double income, DateTime start, DateTime end) {
8          SetIncome(income);
9          SetStart(start);
10         SetEnd(end);
11
12         // Calculate food, living, and entertainment budgets.
13         double food = (1.0/4) * income;
14         double living = (1.0/4) * income;
15         double entertainment = (1.0/4) * income;
16
17         SetFoodBudget(food);
18         SetLivingBudget(living);
19         SetEntertainmentBudget(entertainment);
20         SetSaving();
21     }
22 }
23
```

Input/Output

```
<terminated> NextMonthBudget [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (Nov 8, 2016, 2:24:59 PM)
Please enter your monthly net income

1150

My budget of the next month:
Income: $1150.0
Start date: 2016-12-01
End date: 2016-12-31
Food budget: $575
Living budget: $383
Entertainment budget: $192
Expected saving: $0
```

```
<terminated> NextMonthBudget [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (Nov 8, 2016, 2:26:07 PM)
Please enter your monthly net income

1986

My budget of the next month:
Income: $1986.0
Start date: 2016-12-01
End date: 2016-12-31
Food budget: $497
Living budget: $497
Entertainment budget: $497
Expected saving: $497
```

Conclusion

- Definition
- Benefits
- Negatives
- Illustration
- UML
- Code Example
- Input/Output of Code
- Questions

Questions/Compliments