PareDox
Education Solutions

Aamir Shabbir Pare

Problem Solving Programming

Design Patterns

PareDox
Education Solutions

Recap of Previous Lecture

Abstract Factory Design Pattern

Aamir Shabbir Pare

PareDox
Education Solutions

Abstract Factory Pattern Concept

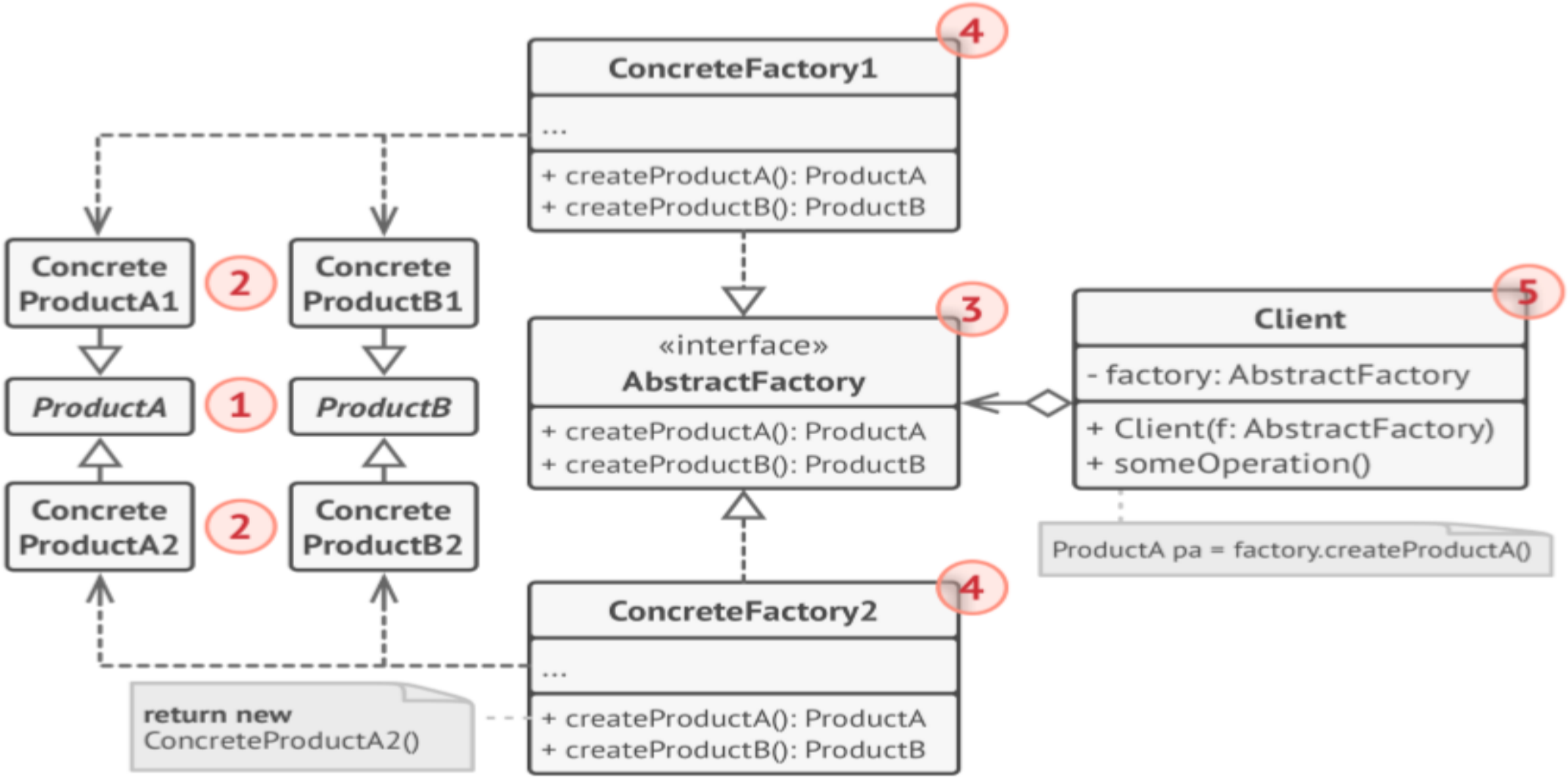
GoF Definition

- Abstract Factory is a creational design pattern that lets you produce families of related objects without specifying their concrete classes.
- In this pattern, you provide a way to encapsulate a group of individual factories that have a common theme.
- This pattern helps you to interchange specific implementations without changing the code that uses them, even at runtime. However, it may result in unnecessary complexity and extra work.
- An abstract factory is called a factory of factories.

Aamir Shabbir Pare

PareDox
Education Solutions

Abstract Factory Pattern Structure



Aamir Shabbir Pare

PareDox
Education Solutions

Disadvantages

- Code is complex.
- A lot of understanding is required.
- Difficult to implement.
- The most complex in creational patterns.
- Refactoring – Factory Method.
- Pattern within a pattern.

Video Lecture Link : http://203.124.40.59:5802/Videos

Aamir Shabbir Pare

PareDox
Education Solutions

Design Patterns Lecture - 16

Builder Pattern

- What is builder design pattern
- Implementation and Guideline
- Practical Implementation using C#

Aamir Shabbir Pare

PareDox
Education Solutions

Introduction

- The Builder Pattern is a common software design pattern that's used to encapsulate the construction logic for an object.
- This pattern is often used when the construction process of an object is complex.
- It's also well suited for constructing multiple representations of the same class.
- Let us first, go over the basics of the pattern, and then with code examples crystallize the concepts.

Aamir Shabbir Pare

PareDox
Education Solutions

What is Builder Design Pattern

The Concept

- **Builder** is a creational design pattern that lets you construct complex objects step by step. The pattern allows you to produce different types and representations of an object using the same construction code.
- Gang of Four Definition say:
  - Separates the construction of a complex object from its representation so that the same construction process can create different representations.
- Solves the problem of increasing constructor parameters and constructors of a given class by providing a step-by-step initialization of parameters.
- After step-by-step initialization, it returns the constructed object at once.

Aamir Shabbir Pare

PareDox
Education Solutions

Wikipedia:

- The builder pattern is a design pattern designed to provide a flexible solution to various object creation problems in object-oriented programming. The intent of the Builder design pattern is to separate the construction of a complex object from its representation. It is one of the Gang of Four design patterns.

Aamir Shabbir Pare

PareDox
Education Solutions

Implementation Guidelines

Choose builder pattern when:

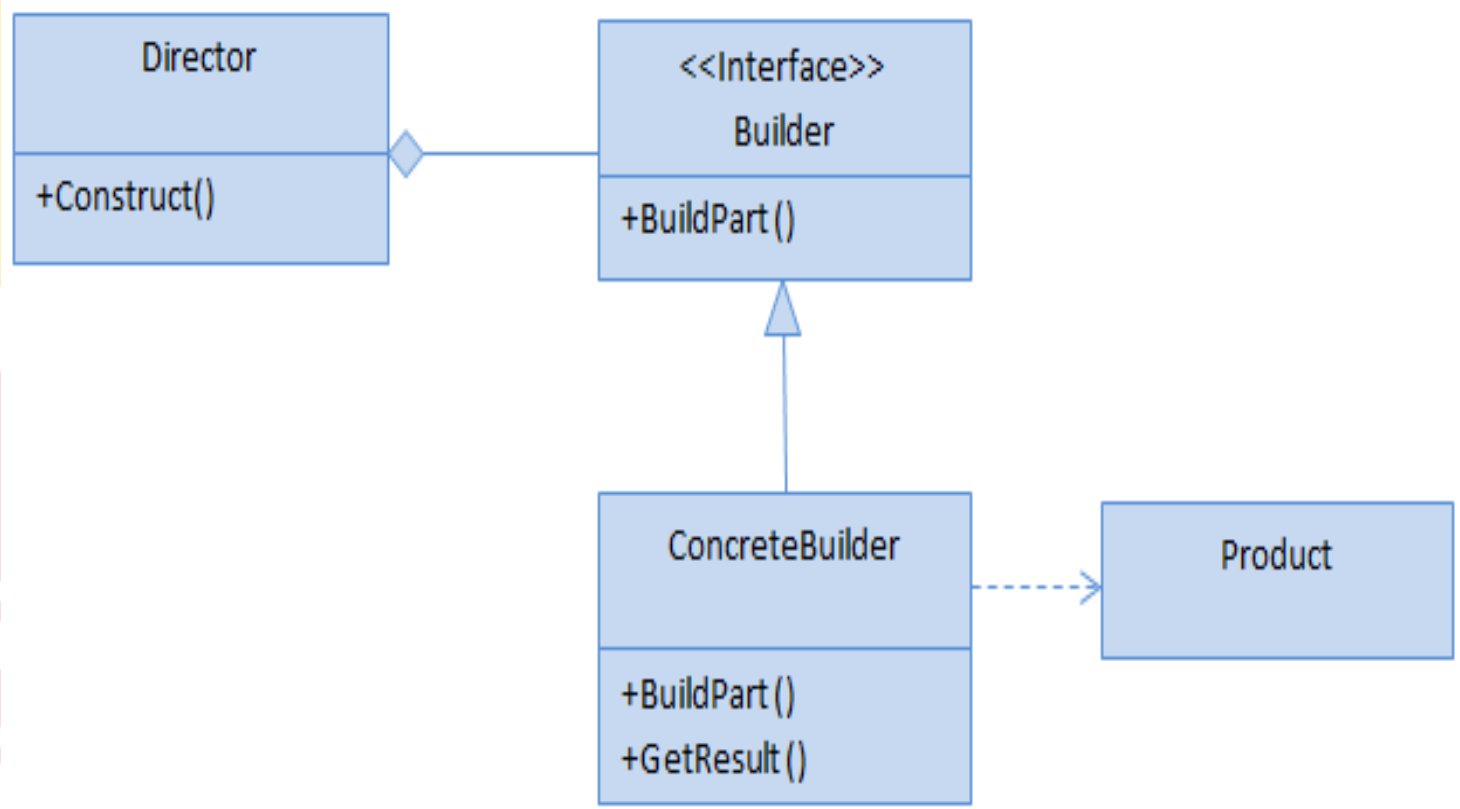
- Need to breakup the construction of a complex object
- Need to build a complex object part-wise independently.
- Construction process must allow multiple representations of the same class.

Aamir Shabbir Pare

Builder Pattern Structure

The Structure

- Builder Interface
- Concrete Builder
- Product
- Director



Builder Pattern

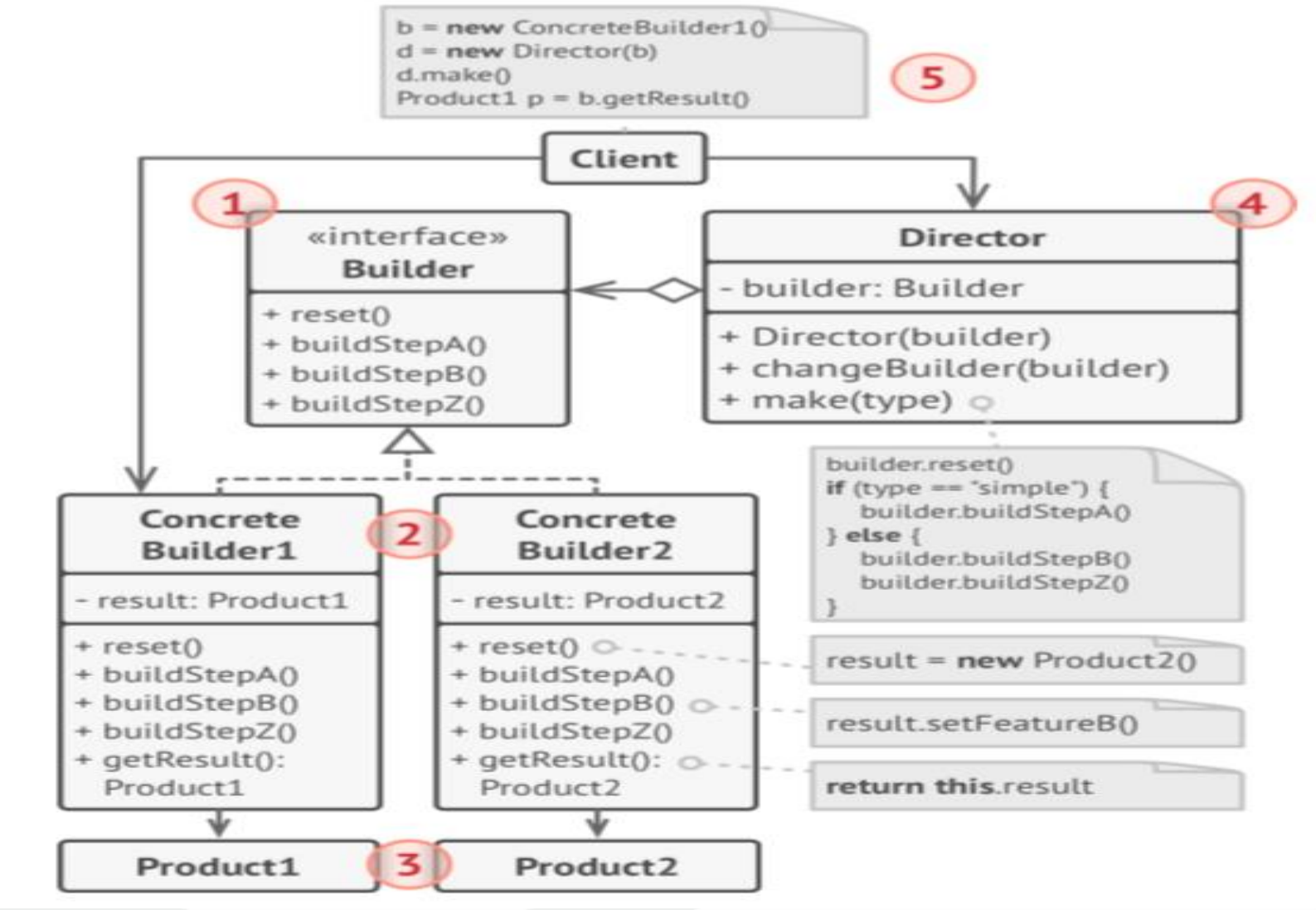
Aamir Shabbir Pare

PareDox
Education Solutions

Builder Pattern Structure

The Structure

- Building Multiple Products Structure



Aamir Shabbir Pare

PareDox
Education Solutions

Participants of Builder Pattern

Builder

This is an interface which is used to define all the steps to create a product

Concrete Builder

This is a class which implements the Builder interface to create a complex product.

Product

This is a class which defines the parts of the complex object which are to be generated by the builder pattern.

Director

This is a class which is used to construct an object using the Builder interface.

Aamir Shabbir Pare

PareDox
Education Solutions

Problem

- Imagine a complex object that requires a laborious, step-by-step initialization of many fields and nested objects.
- Monstrous constructor with lots of parameters.
- Scattered all over the client code.
- How to build a simple house?

Aamir Shabbir Pare

PareDox
Education Solutions

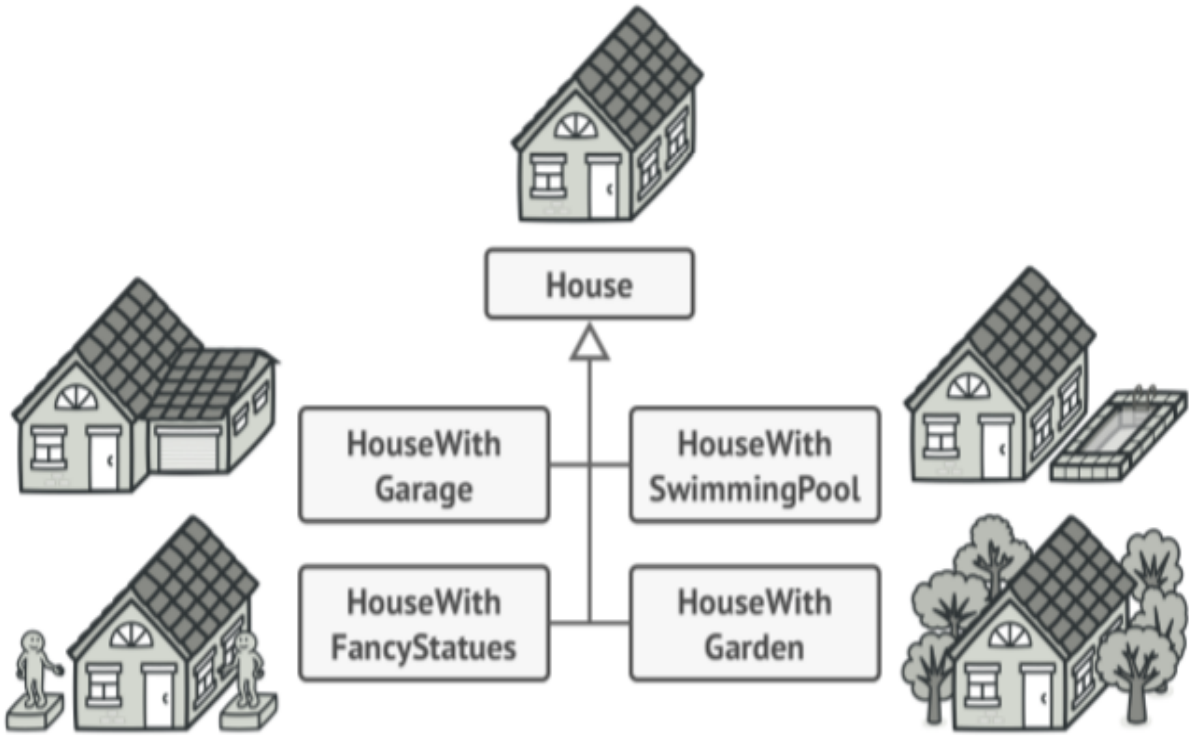
Problem

- How to build a simple house?
- Four walls, a floor, install a door, fit a pair of windows, and build a roof.
- What about bigger house with backyard and swimming pool?
- What if we need a house with other representation like heating system, plumbing and networking cable installation?

Aamir Shabbir Pare

Solution using Inheritance

- A base class House
- Create other subclasses that will cover all combinations of parameters.
- This approach seems to end up with considerable number of subclasses.

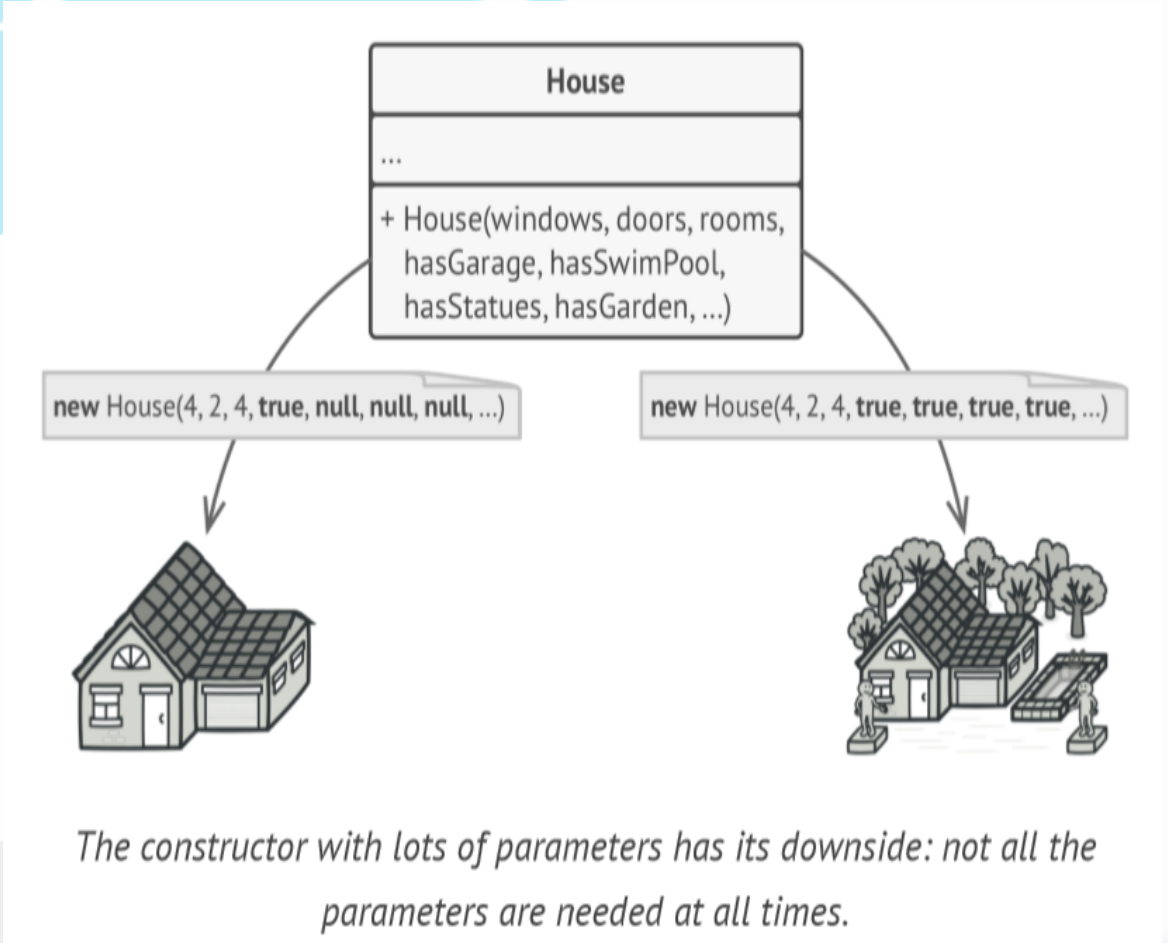


You might make the program too complex by creating a subclass for every possible configuration of an object.

Aamir Shabbir Pare

Solution using Method Parameters

- Create a giant constructor right in the base class.
- Provide all possible parameters in the constructor that control the house object.
- Breeding subclasses will be solved but another problem is born.
- In many cases most of the parameters will be unused.
- For instance, only a fraction of houses have swimming pools, so the parameters related to swimming pools will be useless nine times out often.



The constructor with lots of parameters has its downside: not all the parameters are needed at all times.

PareDox
Education Solutions

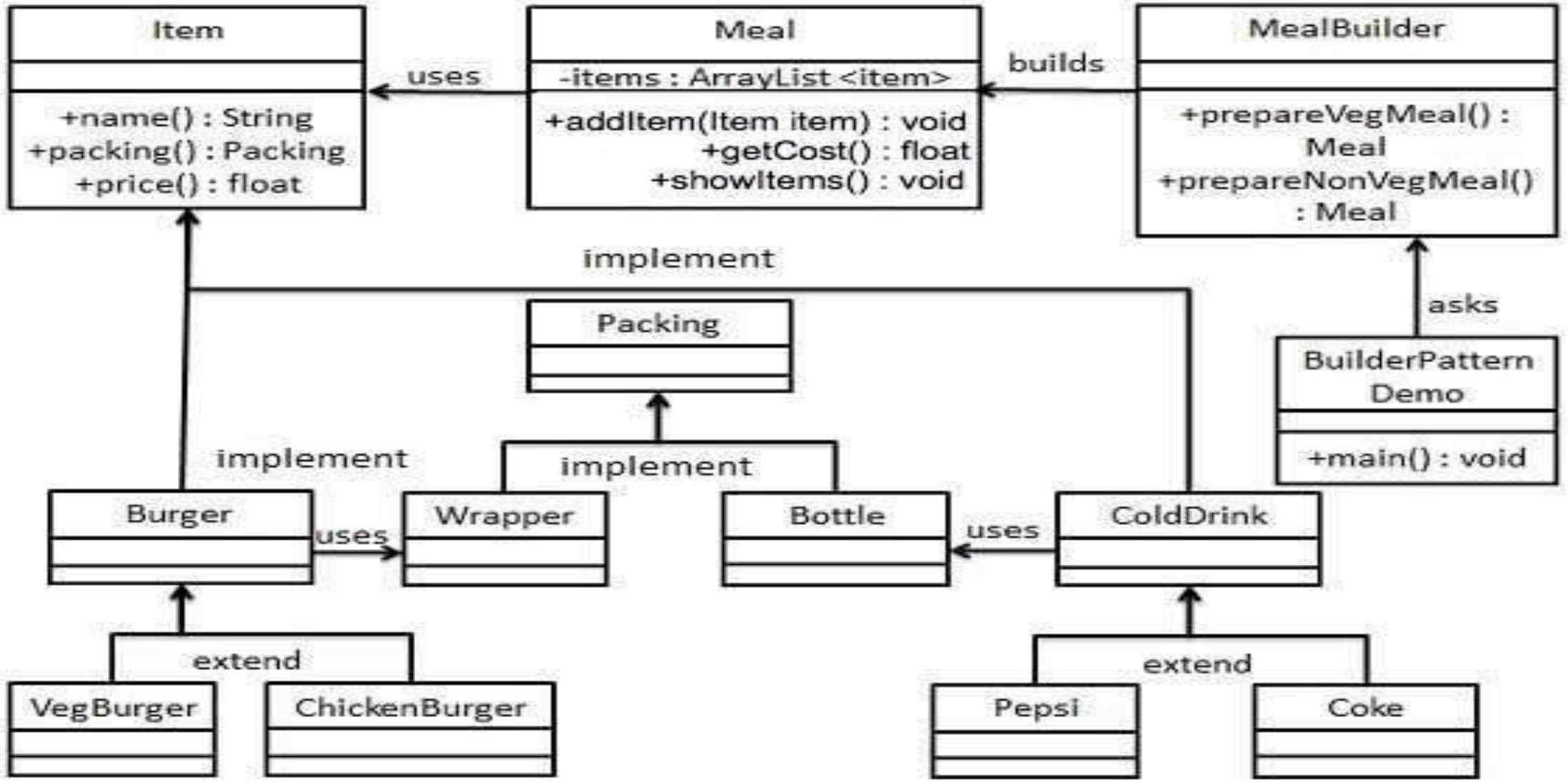
Fast Food Restaurant Business Case

- A typical meal could be a burger and a cold drink.
- Burger could be either a Veg Burger or Chicken Burger and will be packed by a wrapper.
- Cold drink could be either a coke or Pepsi and will be packed in a bottle.
- *Item* interface represents food items such as burgers and cold drinks and concrete classes implement the *Item* interface and a *Packing* interface represents packaging of food items and concrete classes implement the *Packing* interface as burger would be packed in wrapper and cold drink would be packed as bottle.
- *Meal* class have *ArrayList* of *Item* and a *MealBuilder* to build different types of *Meal* objects by combining *Item*.
- *BuilderPatternDemo*, demo class will use *MealBuilder* to build a *Meal*.

PareDox
Education Solutions

Fast Food Restaurant Business Case

Structure



Aamir Shabbir Pare