CS3300 Introduction to Software Engineering

Lecture 13: Design Patterns

Nimisha Roy ▸ nroy9@gatech.edu

Slides adapted from Alessandro Orso

History of Design Patterns

1977
Christopher Alexander introduces the idea of patterns: successful solutions to problems

1987
Ward Cunningham and Kent Beck leverage Alexander’s idea in the context of OO language

1987
Erich Gamma’s dissertation on importance of patterns and how to capture them

1992
Jim Coplien’s book on *Advanced C++ Programming styles and idioms*

Slide adapted from Alessandro Orso

History of Design Patterns

Erich Gamma
Richard Hem
Ralph Johnson
John Vlissides
(gang of four)

Book “Design Patterns: Elements of Reusable OO Software”

Slide adapted from Alessandro Orso

Patterns Catalogue

Fundamental Patterns

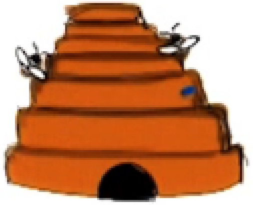
- Delegation pattern
- Interface pattern
- Proxy pattern
- …

Creational Patterns

- Abstract Factory pattern
- Factory Method pattern
- Lazy Initialization pattern
- Singleton pattern
- …

Structural Patterns

- Adapter pattern
- Bridge pattern
- Decorator pattern
- …



Behavioral Patterns

- Chain of responsibility pattern
- Iterator pattern
- Observer pattern
- State Pattern
- Strategy pattern
- Visitor pattern
- …

Concurrency Patterns

- Active object pattern
- Monitor object pattern
- Thread pool pattern
- …

Slide adapted from Alessandro Orso

Format (Subset)

Name
Intent
Motivation
Applicability
Structure
Consequences
Implementation
Sample Code
Related Patterns

Slide adapted from Alessandro Orso

Factory Method Pattern- Intent & Applicability

Intent

Allows for creating objects without specifying their class, by invoking a factory method (i.e., a method whose main goal is to create class instances)

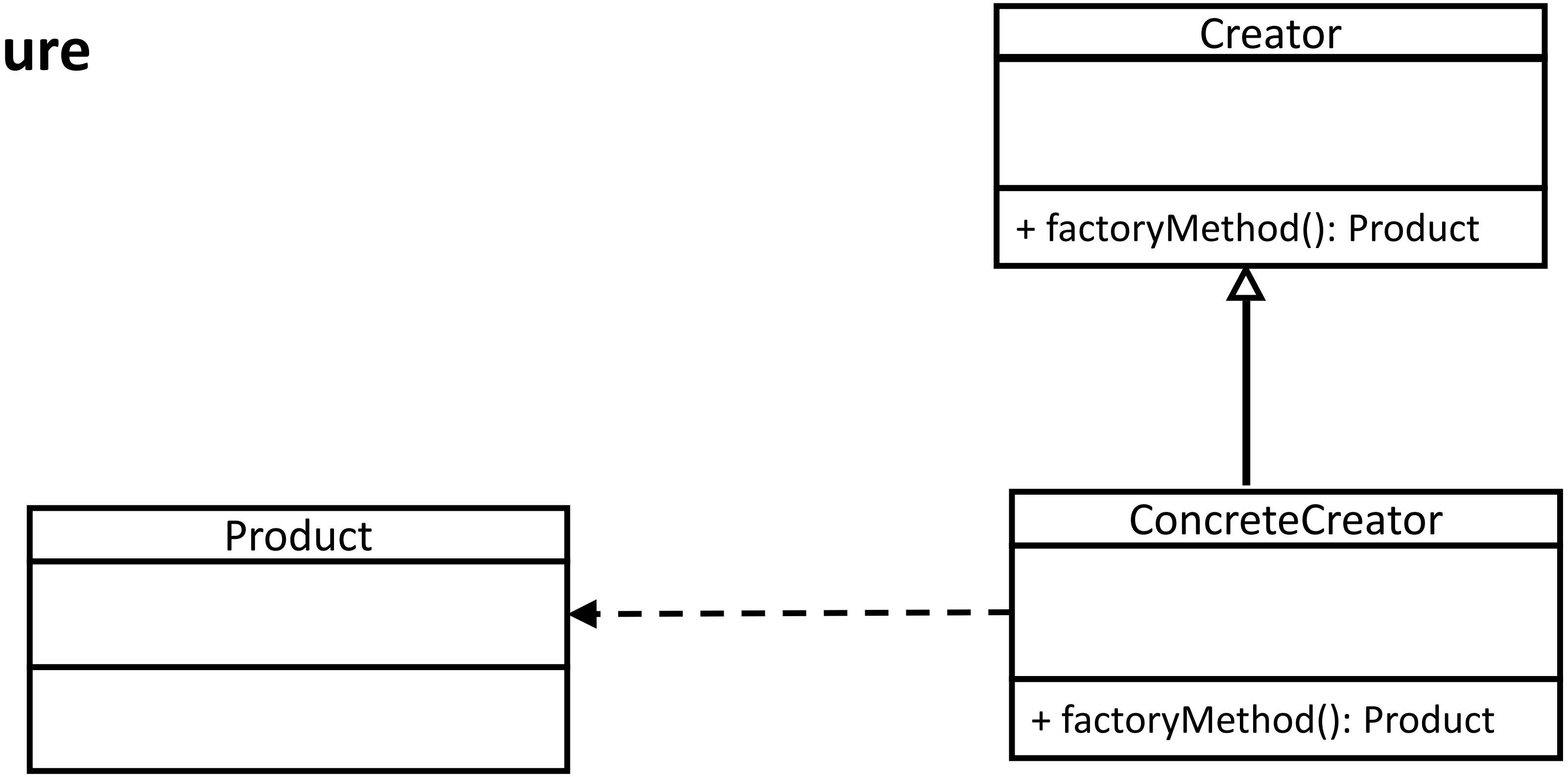
Applicability

- Class can’t anticipate the type of objects it must create
- Class wants its subclasses to specify the type of objects it creates
- Class needs control over the creation of its objects

Slide adapted from Alessandro Orso

Factory Method Pattern- Structure & Participants

Structure



Participants

Creator: provides interface for factory method

ConcreteCreator: provides method for creating actual object

Product: Object created by the factory method

Slide adapted from Alessandro Orso

Factory Method Pattern - Sample Code

```
public class ImageReaderFactory{
 public static ImageReader createImageReader (InputStream is){
      int imageType = getImageType(is);
      switch(imageType){
       case ImageReaderFactory.GIF
        return new GifReader (is);
       case ImageReaderFactory.JPEG
        return new JpegReader (is);
       }
      }
    }
```

Some other examples and implementation of factory method patterns in Java can be found here and here.

Slide adapted from Alessandro Orso

Strategy Pattern- Intent & Applicability

Intent

Allows for switching between different algorithms for accomplishing a task

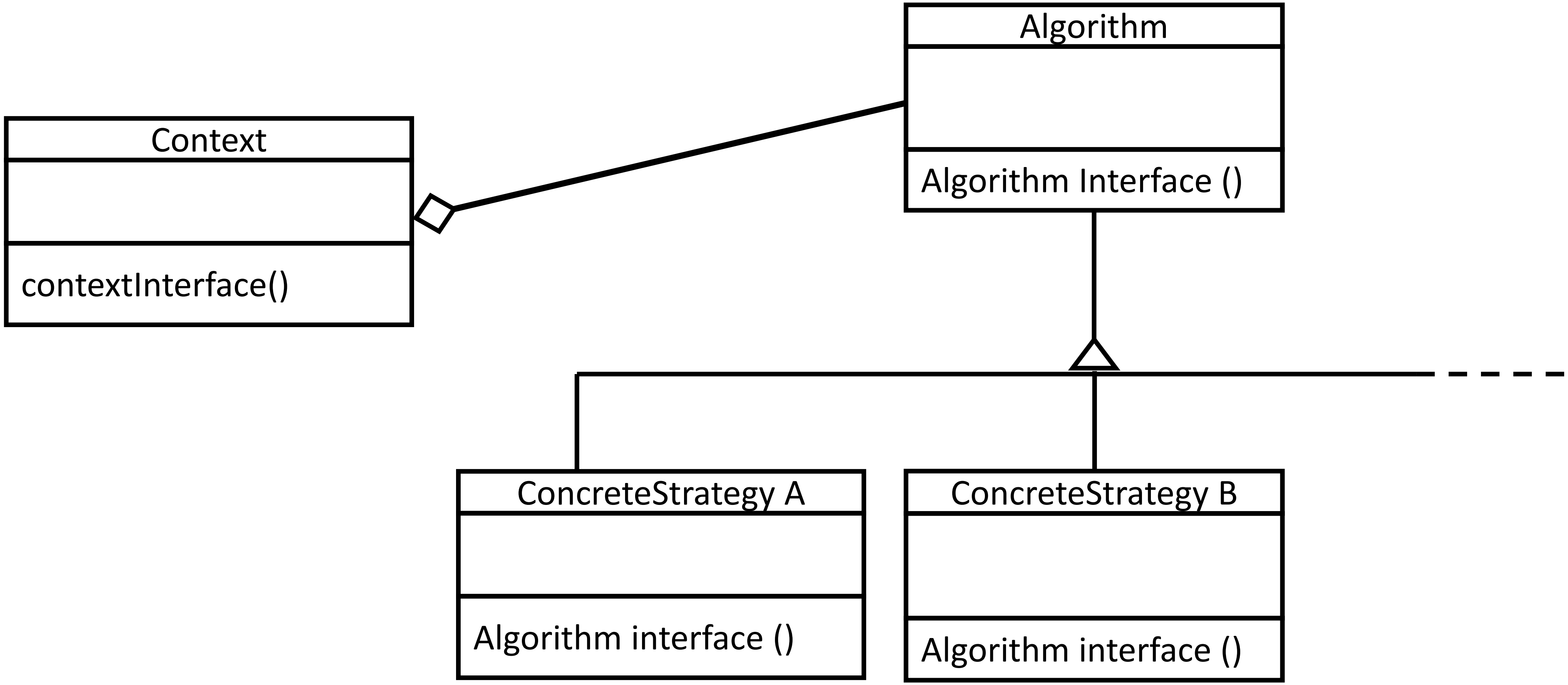
Applicability

- Different variants of an algorithm
- Many related classes differ only in their behavior

Slide adapted from Alessandro Orso

Strategy Pattern- Structure & Participants

Structure



Participants

Context: provides interface to outside world

Algorithm (strategy): common interface for the different algorithms

Concrete Strategy: actual implementation of the algorithms

Slide adapted from Alessandro Orso

Strategy Pattern: Example

Program

- Input: Text file
- Output: Filtered File

Four filters

- No filtering
- Only words that start with “t”
- Only words longer than 5 characters
- Only words that are palindromes

Slide adapted from Alessandro Orso

Strategy Pattern

Example Demo

Slide adapted from Alessandro Orso

Other Common Patterns

Visitor: A way of separating an algorithm from an object structure on which it operates (Example)

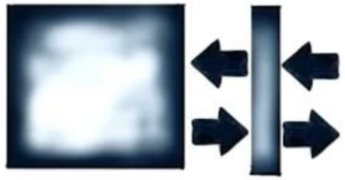
Decorator: A wrapper that adds functionality to a class: stackable (Example)

Iterator: Access elements of a collection without knowing underlying representation (Example)

Slide adapted from Alessandro Orso

Other Common Patterns

Observer: Notify dependents when object of interest changes



Proxy: Surrogate controls access to an object

Slide adapted from Alessandro Orso

Choosing a Pattern

Approach

- Understand your design context
- Examine the patterns catalogue
- Identify and study related patterns
- Apply suitable pattern

Pitfalls

- Selecting wrong patterns
- Abusing patterns

Slide adapted from Alessandro Orso

Were you paying attention?

Imagine that you have to write a class that can have one instance only. Using one of the design patterns that we discussed in this lesson, write the code of a class with only one method (except for possible constructors) that satisfies this requirement. Make sure to call the class Singleton.

What pattern should be followed?
Factory

```
public class Singleton {
		private static Singleton instance;
		private Singleton() {}
		public static Singleton factory() {
				if (instance == null) {
						instance = new Singleton();
				}
				return instance;
		}
}
```

Negative Design Patterns

Also in Christopher Alexander’s book

How not to (design, manage, etc.)

Also called anti-patterns and bad smells

Slide adapted from Alessandro Orso