# Design Pattern

Herdi Ashaury

# Design Pattern

Design Pattern (Pola desain) mewakili praktik terbaik yang digunakan oleh pengembang perangkat lunak berorientasi objek yang berpengalaman. Design Pattern adalah solusi untuk masalah umum yang dihadapi pengembang perangkat lunak selama pengembangan perangkat lunak. Solusi ini diperoleh secara coba-coba oleh banyak pengembang perangkat lunak selama periode waktu yang cukup substansial.

# Gang of Four (GoF)

Pada tahun 1994, empat penulis Erich Gamma, Richard Helm, Ralph Johnson dan John Vlissides menerbitkan sebuah buku berjudul **Design Patterns - Elements of Reusable Object-Oriented Software** yang memprakarsai konsep Design Pattern dalam pengembangan perangkat lunak.

Para penulis ini secara kolektif dikenal sebagai Gang of Four (GOF). Menurut para penulis ini design pattern terutama didasarkan pada prinsip-prinsip desain berorientasi objek berikut.

- Program to an Interface bukan implementasi
- Menyukai komposisi objek daripada pewarisan

# Gang of Four (GoF)

Ralph Johnson, Richard Helm, Erich Gamma, John Vlissides

# GoF (Gang of inFormatika)

# Penggunaan Design Pattern

Design Pattern memiliki dua penggunaan utama dalam pengembangan perangkat lunak.

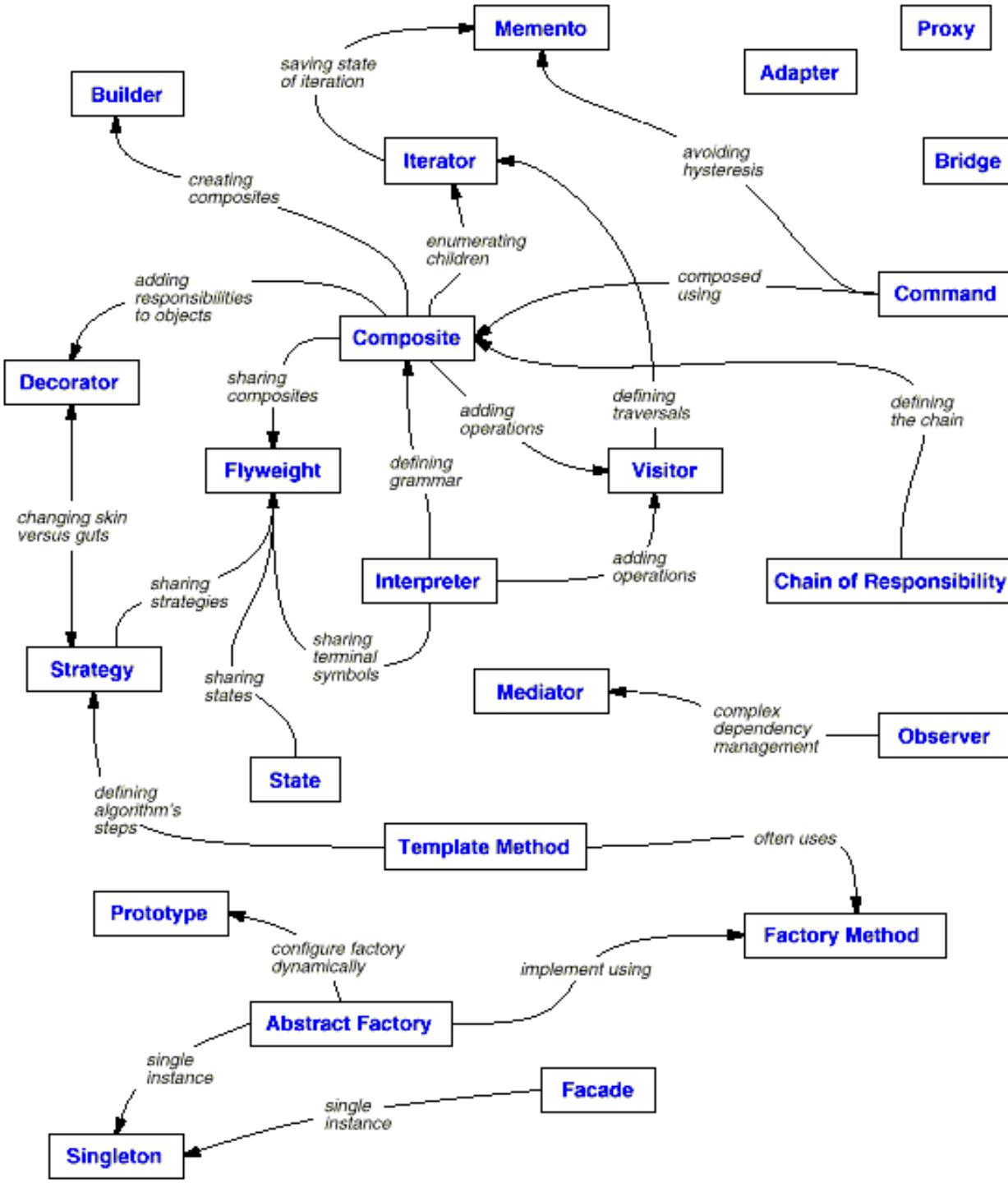
- Platform umum untuk pengembang
- Best Practices

# Kelompok Design Pattern

Dalam buku dari GoF, terdapat 23 Design Pattern. Pola-pola ini dapat diklasifikasikan ke dalam tiga kategori:

- Creational Patterns (Pola penciptaan)
- Struktural Patterns
- Behavioral Patterns

# GoF Pattern Summary & Relationships



# Creational Patterns

Pola desain ini menyediakan cara untuk membuat objek sambil menyembunyikan logika pembuatan, daripada membuat instance objek langsung menggunakan operator baru. Ini memberi program lebih banyak fleksibilitas dalam memutuskan objek mana yang perlu dibuat untuk use case yang diberikan.

1. **Factory Pattern**
2. Abstract Factory Pattern
3. **Singleton Pattern**
4. Prototype Pattern
5. **Builder Pattern**.

# Structural Patterns

Pola desain ini menyangkut komposisi kelas dan objek. Konsep pewarisan digunakan untuk menyusun antarmuka dan menentukan cara untuk menyusun objek untuk mendapatkan fungsionalitas baru.

1. **Adapter Pattern**
2. Bridge Pattern
3. Composite Pattern
4. Decorator Pattern
5. **Facade Pattern**
6. Flyweight Pattern
7. Proxy Pattern

# Behavioral Patterns

Pola desain ini secara khusus berkaitan dengan komunikasi antara objek.

1. Chain Of Responsibility Pattern
2. Command Pattern
3. Interpreter Pattern
4. Iterator Pattern
5. Mediator Pattern
6. Memento Pattern
7. **Observer Pattern**
8. **State Pattern**
9. **Strategy Pattern**
10. Template Pattern
11. Visitor Pattern

# Factory Pattern

Factory Pattern atau Factory Method Pattern yaitu suatu pola yang mendefinisikan **Interface** atau **Abstract Class** tetapi menggunakan **subclass (child)** untuk membuat objek yang akan dipakai.

Dengan kata lain, subclass bertanggung jawab untuk membuat instance kelas.

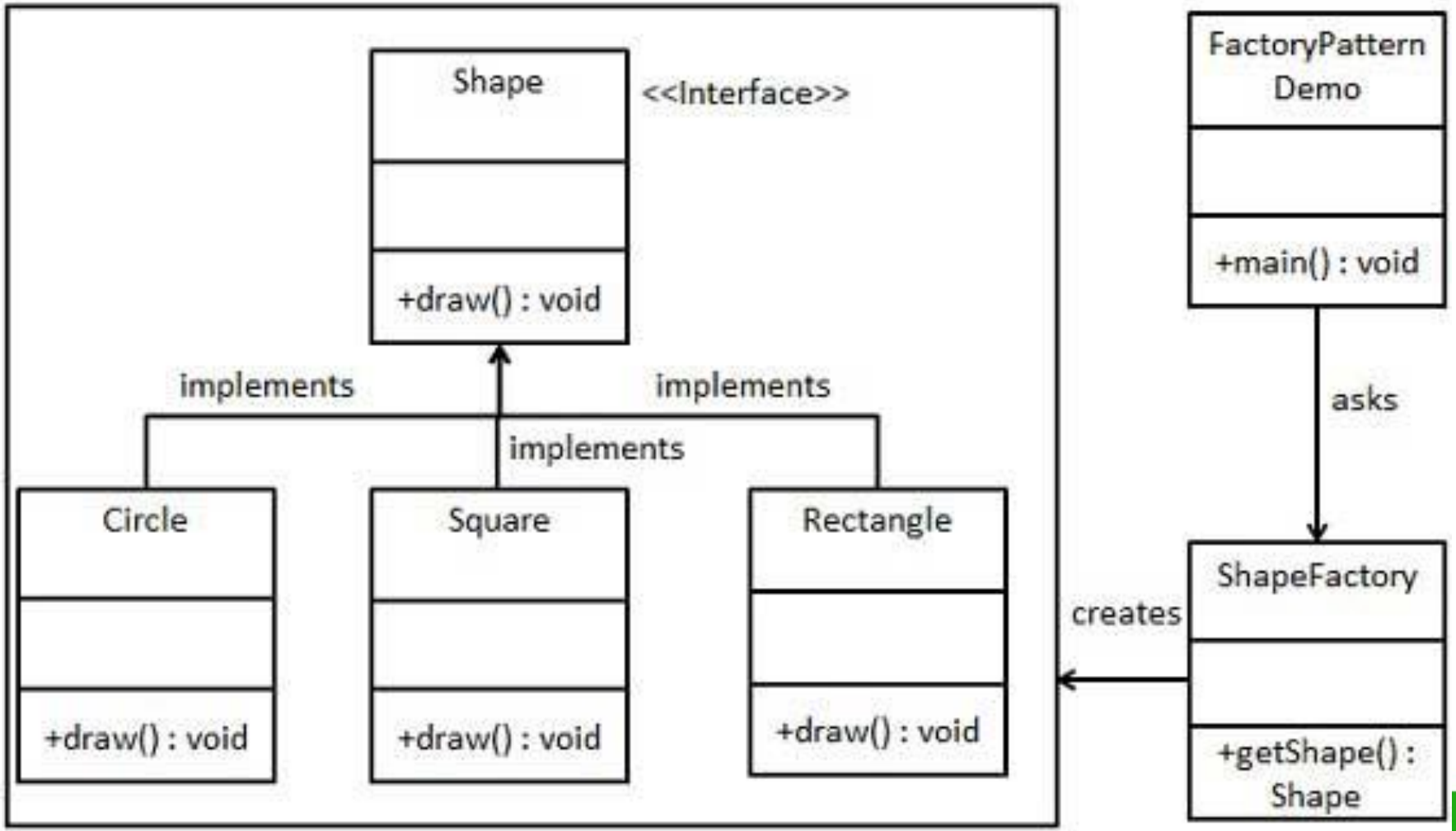
Factory Method Pattern juga dikenal sebagai **Virtual Constructor**.

# Factory Pattern

Provide an interface for creating families of related or dependent objects without specifying their concrete classes.

- Control instantiation
- Singleton is a special case of Factory where only one object can be created.

# Factory Pattern Implementation



**Shape.java**

```
public interface Shape {
   void draw();
}
```

**Square.java**

```
public class Square implements Shape {
   @Override
   public void draw() {
      System.out.println("Inside
Square::draw() method.");
   }
}
```

**FactoryPatternDemo.java**

```
public class FactoryPatternDemo {
   public static void main(String[]
args) {
      ShapeFactory shapeFactory = new
ShapeFactory();
      //get an object of Square and
call its draw method.
      Shape shape1 =
shapeFactory.getShape(“Square");
      //call draw method of Circle
      shape1.draw();
        }
}
```

# Singleton Pattern

Kelas yang hanya dapat diinstansiasi menjadi 1 objek.

Setiap yang membutuhkan layanan kelas ini akan mendapatkan instans objek yang sama.

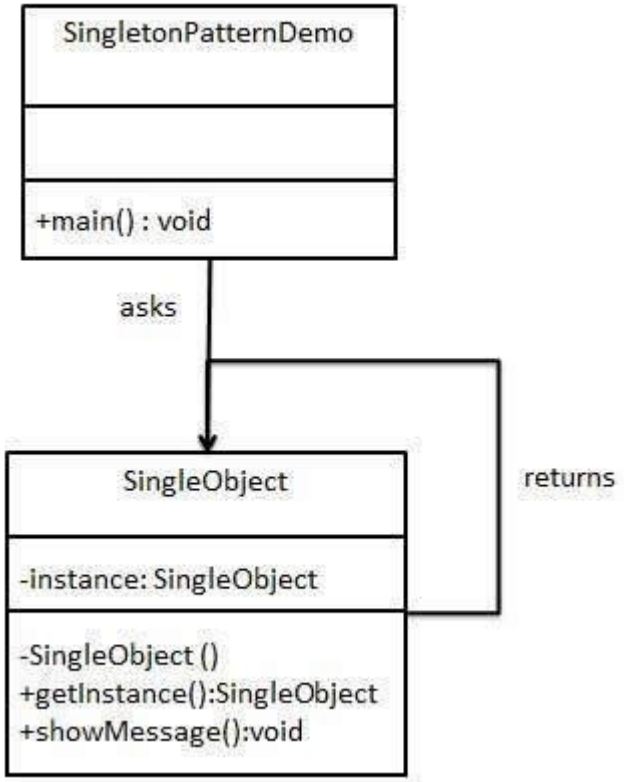
Kelas ini memiliki konstruktor yang hak aksesnya adalah private (atau protected dalam beberapa kasus).

Konstruktor hanya dapat dipanggil dari dalam kelas.

Objek lain yang membutuhkan kelas ini dapat meminta instans kelas dengan memanggil method statik GetInstance.

# Singleton Pattern

```
public class SingleObject {

   //create an object of SingleObject

   private static SingleObject instance = new
SingleObject();

   //make the constructor private so that this class
cannot be

   //instantiated

   private SingleObject(){}

   //Get the only object available

   public static SingleObject getInstance(){

      return instance;

   }

   public void showMessage(){

      System.out.println("Hello World!");

   }

}
```



# Referensi


- https://www.tutorialspoint.com/design_pattern/design_pattern_quick_guide.htm
- https://www.javatpoint.com/design-patterns-in-java