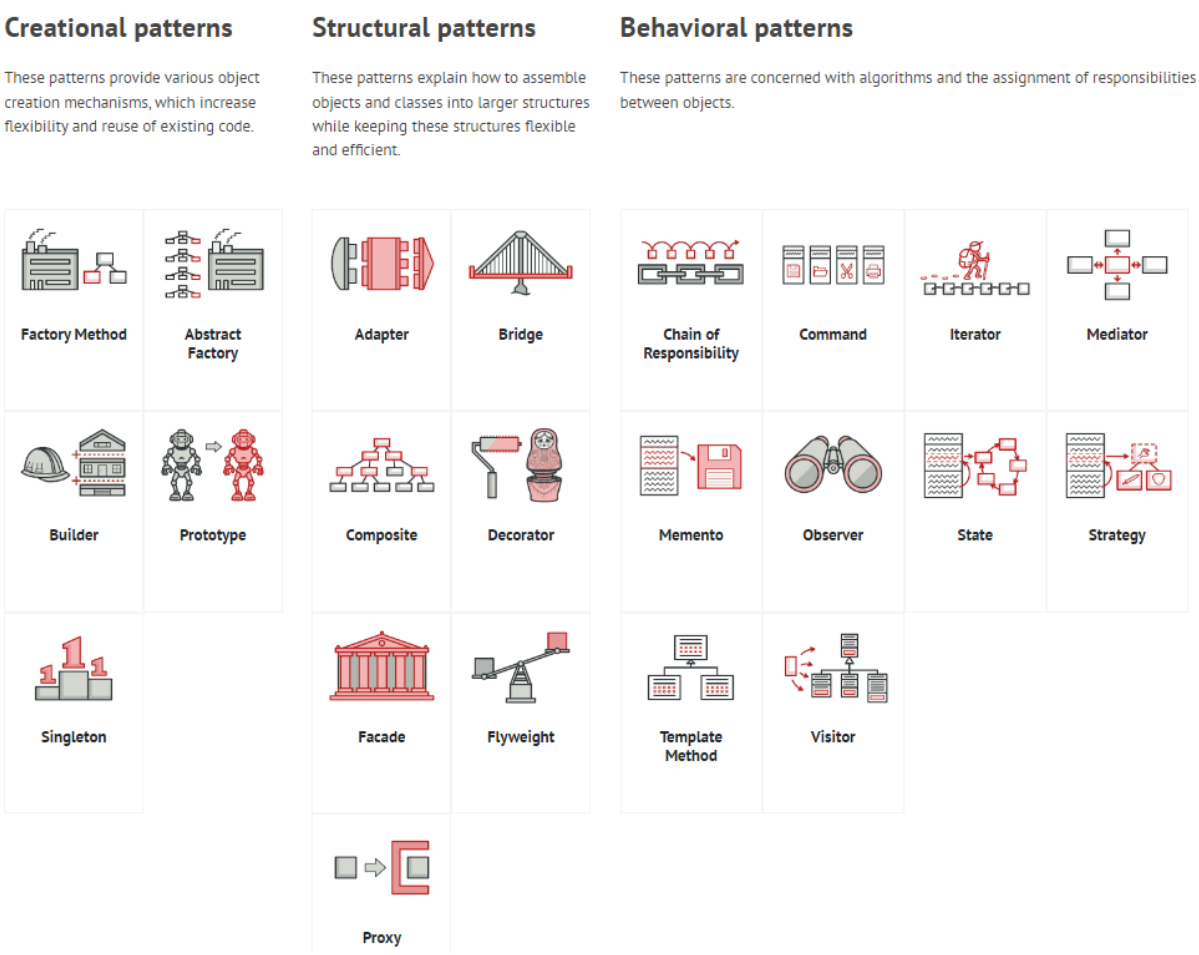
Design Patterns

Design Patterns: Design patterns are typical solutions to commonly occurring problems in software design. They are like pre-made blueprints that you can customize to solve a recurring design problem in your code.

They were first introduced in the book **Design Patterns: Elements of Reusable Object-Oriented Software**, published in 1994. The book was written by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, collectively known as **Gang of Four**. The design patterns in this book are also commonly known as **GoF design patterns**.

Design Pattern Types



Singleton

Singleton is a creational design pattern that lets you ensure that a class has only one instance, while providing a global access point to this instance.

The Singleton pattern solves two problems at the same time

1. Ensure that a class has just a single instance
2. Provide a global access point to that instance

Solution

1. Make the default constructor private, to prevent other objects from using the `new` operator with the Singleton class.
2. Create a static creation method that acts as a constructor.

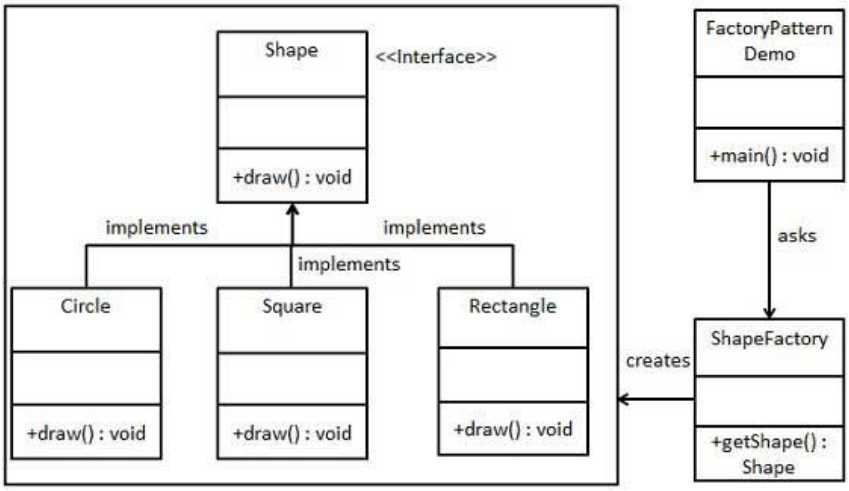
```
class Singleton {

        private static Singleton ins = null;
        private Singleton(){}
        public static Singleton getInstance()
        {
                if (ins == null)
                ins = new Singleton();

                return ins;
        }
}
```

```
class GFG {
        public static void main(String args[])
        {
                Singleton x = Singleton.getInstance();
                Singleton y = Singleton.getInstance();
                Singleton z = Singleton.getInstance();
                System.out.println("Hashcode of x is "
                                                + x.hashCode());
                System.out.println("Hashcode of y is "
                                                + y.hashCode());
                System.out.println("Hashcode of z is "
                                                + z.hashCode());
        }
}
```

Output:
Hashcode of x is 558638686
Hashcode of y is 558638686
Hashcode of z is 558638686

Factory

The Factory Design Pattern is a creational design pattern that provides an interface for creating objects in a super class but allows subclasses to alter the type of objects that will be created.




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

```
public class Rectangle implements Shape {

   @Override
   public void draw() {
      System.out.println("Inside Rectangle::draw() method.");
   }
}
```

```
public class Square implements Shape {

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

```
public class ShapeFactory {
   public Shape getShape(String shapeType){
      if(shapeType == null){
         return null;
      }

      if(shapeType.equalsIgnoreCase("RECTANGLE")){
         return new Rectangle();

      } else if(shapeType.equalsIgnoreCase("SQUARE")){
         return new Square();
      }

      return null;
   }
}
```

```
public class FactoryPatternDemo {

   public static void main(String[] args) {
      ShapeFactory shapeFactory = new ShapeFactory();
      Shape shape1 = shapeFactory.getShape("CIRCLE");shape1.draw();
      Shape shape2 = shapeFactory.getShape("RECTANGLE");shape2.draw();
   }
}
```

Output

Inside Circle::draw() method.

Inside Rectangle::draw() method.