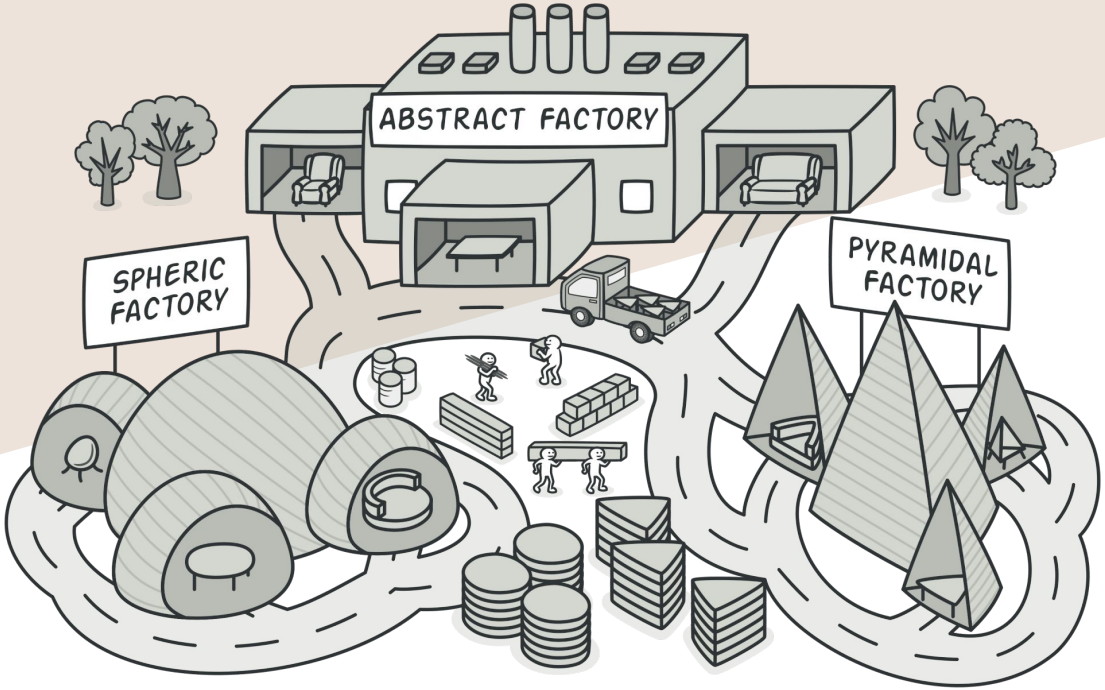
Factory Patterns

By
Jake Turner (100762152)
Riley Dunn (100760858)

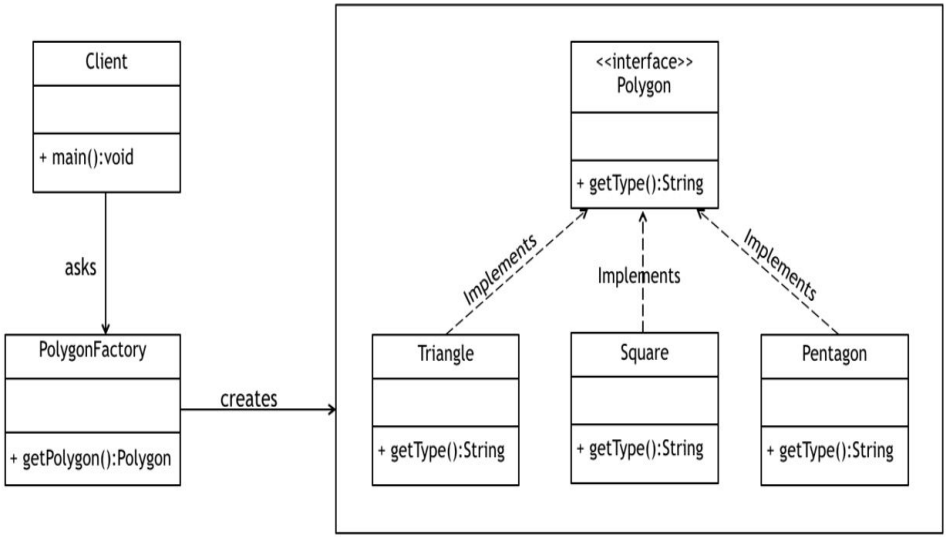


Design Patterns are...

Design Patterns are the best practices for object oriented programming that have been established over time; they are solutions for common problems during software development

Creational	Structural	Behavioral
Factory	Adapter	Interpreter
Abstract Factory	Bridge	Template Method
Builder	Composite	Chain of Responsibility
Prototype	Decorator	Command
Singleton	Facade	Iterator
	Flyweight	Mediator
	Proxy	Memento
		Observer
		State
		Strategy
		Visitor

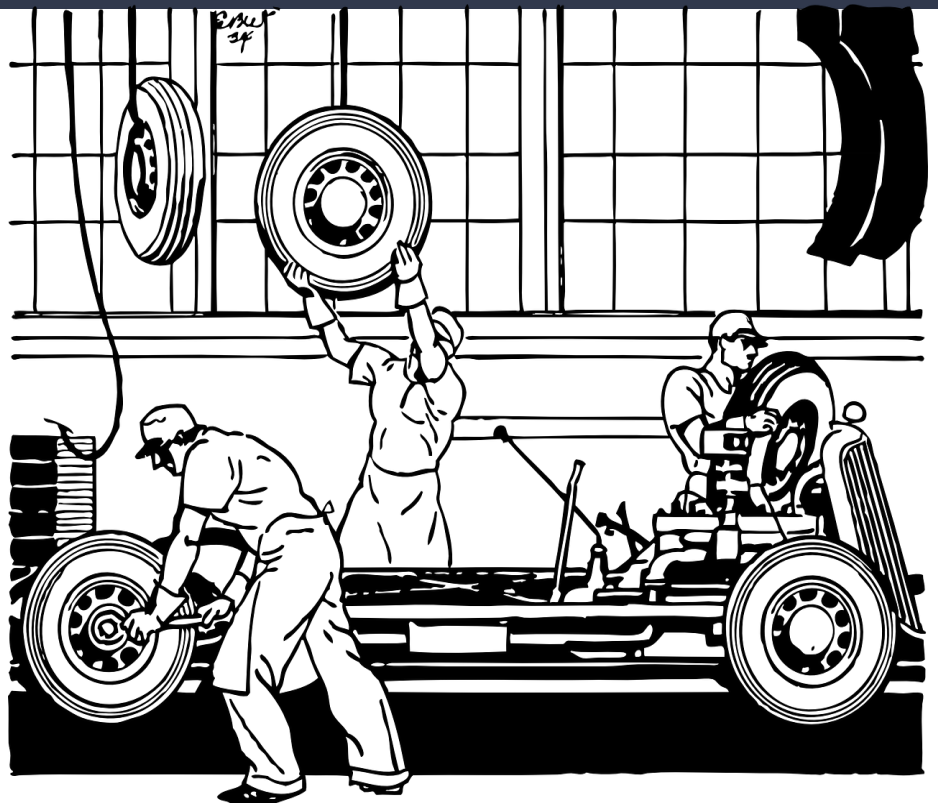
What is a Factory Pattern?



A Factory Pattern is a design pattern that allows you create objects in a superclass, it also gives subclasses the ability to change the type of objects when creating by making subclasses abstract.

When to Use Factory Patterns

Use a Factory pattern when trying to localize the creation of an object. It also allows you to create an object without needing to know what type you need before making them.



When Not to Use Factory Patterns

Only use Factory Patterns when necessary, such as when many abstract objects are required. Factory Patterns should be limited because they cause added complication.

Classes Involved In Factory Pattern

Main class - Starts program and calls related classes

Parent Class - A class that creates other subclasses based on certain parameters

Subclass - are the individual object classes created with parameters set by the parent class

Interface Class

```java
public interface InterfaceClass {
    void StarterC(String starter);
}
```

Main Class

-asks user for starter pokemon and sends user's input to parent class

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);
        ParentClass fC = new ParentClass();


        System.out.println("Chose your starter");
        System.out.println("bulbasaur");
        System.out.println("chimchar");
        System.out.println("piplup");


        String starter = scan.nextLine();
        InterfaceClass cStarter = ParentClass.ChoseStarter(starter);
    }
}
```

Parent Class

-creates subclasses based on info taken from main class

```
public class ParentClass {
    public static InterfaceClass ChoseStarter(String starter){
        if(starter.equalsIgnoreCase( anotherString: "bulbasaur")){
            return new ObjectClass();

        }else if(starter.equalsIgnoreCase( anotherString: "chimchar")) {
            return new ObjectClass2();

        }else if(starter.equalsIgnoreCase( anotherString: "piplup")){
            return new ObjectClass3();

        }else{
            System.out.println("E R R O R");
            return null;
        }
    }
}
```

Object Classes

-individual classes with actions based on specific class
-pokemon will have their own announcements to the user

```
public class ObjectClass implements InterfaceClass{
    public void StarterC(String starter){
        System.out.println("You chose Bulbasaur");
    }
}
public class ObjectClass2 implements InterfaceClass{
    public void StarterC(String starter){
        System.out.println("You chose Chimchar");
    }
}
public class ObjectClass3 implements InterfaceClass{
    public void StarterC(String starter){
        System.out.println("You chose Piplup");
    }
}
```

Summary

-Design patterns are well established best practices for object oriented programing and are a solution to common problems in software development

-The factory pattern method is an easy way to create multiple abstract files

-By changing the superclass, you can easily change how object classes are created

References

Slide 1

- https://refactoring.guru/images/patterns/content/abstract-factory/abstract-factory-en-3x.png

Slide 2

- https://sourcemaking.com/design_patterns
- https://www.researchgate.net/profile/Mohamed-Adel-96/publication/330207932/figure/tbl2/AS:712602380423169@1546909055739/Types-of-design-patterns.png

Slide 3

- https://refactoring.guru/design-patterns/factory-method
- https://www.baeldung.com/wp-content/uploads/2017/11/Factory_Method_Design_Pattern.png

Slide 4

- https://www.sitepoint.com/understanding-the-factory-method-design-pattern/

Slide 5

- https://softwareengineering.stackexchange.com/questions/236141/when-is-it-worth-not-using-a-factory