Design Patterns

Strategy Pattern
Builder Pattern

Strategy Pattern

Strategy is a behavioural design pattern that turns a set of behaviours into objects and makes them interchangeable inside original context object.

```java
import java.util.Collections;
import java.util.List;

interface SortStrategy {
    List<String> doAlgorithm(List<String> data);
}

class AlphabeticSorter implements SortStrategy {
    @Override
    public List<String> doAlgorithm(List<String> data) {
        Collections.sort(data);
        return data;
    }
}

class AlphabeticDescSorter implements SortStrategy {
    @Override
    public List<String> doAlgorithm(List<String> data) {
        Collections.sort(data);
        Collections.reverse(data);

        return data;
    }
}

class Context {
    private SortStrategy strategy;

    public Context(final SortStrategy strategy) {
        this.strategy = strategy;
    }

    public void applySort(SortStrategy strategy) {
        this.strategy = strategy;
    }

    public void performSort(List<String> data) {
        var result = strategy.doAlgorithm(data);
        System.out.println(result);
    }
}
```

Strategy in Java

Strategy is a behavioral design pattern that turns a set of behaviors into objects and makes them interchangeable inside original context object. The original object, called context, holds

https://refactoring.guru/design-patterns/strategy/java/example

Builder Pattern

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.

As an example lets take the nutrition facts of a grocery. The class `NutritionFacts` implements the builder pattern in order to construct an object. Therefore its constructor only accepts a single builder instance. The builder instance is an intermediate cache for the arguments which should be applied to the final `NutritionFacts` instance (while setting all other to their default values).

```
class NutritionFacts {
    private final int servingSize;
    private final int servings;
    private final int calories;
    private final int fat;
    private final int sodium;
    private final int carbohydrate;

    public static class Builder { ... }

    private NutritionFacts(Builder builder) {
        servingSize = builder.servingSize;
        servings = builder.servings;
        calories = builder.calories;
        fat = builder.fat;
        sodium = builder.sodium;
        carbohydrate = builder.carbohydrate;
    }
}
```

The internal `Builder` class looks like the following below. Each method updates the current instance and applies its setting accordingly.

```
public static class Builder {
      // Required parameters
      private final int servingSize;
      private final int servings;
      // Optional parameters - initialized to default values
      private int calories = 0;
      private int fat = 0;
      private int sodium = 0;
      private int carbohydrate = 0;

      public Builder(int servingSize, int servings) {
          this.servingSize = servingSize;
          this.servings = servings;
      }

      public Builder calories(int val) {
          calories = val;
          return this;
      }

      public Builder fat(int val) {
          fat = val;
          return this;
      }

      public Builder sodium(int val) {
          sodium = val;
          return this;
      }

      public Builder carbohydrate(int val) {
          carbohydrate = val;
          return this;
      }

      public NutritionFacts build() {
          return new NutritionFacts(this);
      }
}
```

A `NutritionFacts` instance would be constructed as the following:

```
NutritionFacts cocaCola = new NutritionFacts.Builder(240, 8)
  .calories(100).sodium(35).carbohydrate(27).build();
```

Builder

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

https://refactoring.guru/design-patterns/builder