Builder Design Pattern

Tyler Cruz
Ciara Ryan
Michael Plescia

What is it?

- In General: allows user to construct complex objects step by step
- Used when an object needs multiple steps and could have different variations
  - Step-by-step guide to construct objects
  - Uses a builder code
  - Should only use for building immutable objects
- Helps manage and extend the construction process without modifying the actual product class
- Similar to Abstract Factory Pattern
  - Abstract Factory returns a product immediately and has a family of objects
  - Builder returns and builds one object at the end

Problems it Solves

- Construct complex trees or complex objects
  - Step-by-step allows code to be more flexible
  - Allows the creation of different types of a product using the same construction code
  - Dealing with objects that have numerous optional parameters
- Creating different variations of a product or objects
  - Creation of different types and representations of a product using a base builder
- Telescopic constructors: when a class has multiple constructors, each with a different combination of parameters.
  - Instead of having a ton of constructors, uses a dedicated builder class to set the parameters making the construction process simpler
  - No overloading or cramming parameters into constructors

Pros and Cons

Pros

- Flexible
- Reusable
- Isolatable

Cons

- Complex to write

Builder Example

```java
import java.time.Year;

public class Book {
    private final String isbn;
    private final String title;
    private final Genre genre;
    private final String author;
    private final Year published;
    private final String description;
    public Book(String isbn, String title, Genre genre, String author, Year published, String description) {
        this.isbn = isbn;
        this.title = title;
        this.genre = genre;
        this.author = author;
        this.published = published;
        this.description = description;
    }
```

```java
    public String getIsbn() {
        return isbn;

    }
    public String getTitle() {

        return title;
```

```java

    public Genre getGenre() {
        return genre;
    }

    public String getAuthor() {
        return author;
```

```java
    public Year getPublished() {
        return published;
    }

    public String getDescription() {
        return description;
    }
```

Effective Java Builder Pattern

```java
import java.time.Year;

public class Book {
    private final String isbn;
    private final String title;
    private final Genre genre;
    private final String author;
    private final Year published;
    private final String description;
    private Book(Builder builder) {
        this.isbn = builder.isbn;
        this.title = builder.title;
        this.genre = builder.genre;
        this.author = builder.author;
        this.published = builder.published;
        this.description = builder.description;
    }

    public String getIsbn() {
        return isbn;
    }

    public String getTitle() {
        return title;
    }

    public Genre getGenre() {
        return genre;
    }

    public String getAuthor() {
        return author;
    }

    public Year getPublished() {
        return published;
    }

    public String getDescription() {
        return description;
    }
```

```java
    public static class Builder {
        private final String isbn;
        private final String title;
        private Genre genre;
        private String author;
        private Year published;
        private String description;

        public Builder(String isbn, String title) {
            this.isbn = isbn;
            this.title = title;
        }

        public Builder genre(Genre genre) {
            this.genre = genre;
            return this;
        }

        public Builder author(String author) {
            this.author = author;
            return this;
        }

        public Builder published(Year published) {
            this.published = published;
            return this;
        }

        public Builder description(String description) {
            this.description = description;
            return this;
        }

        public Book build() {
            return new Book(this);
        }

    }

}
```

Utilization

```
Book book = new Book.Builder(isbn:"0-12-345678-9", title:"Moby-Dick")
            .genre(Genre.ADVENTURE_FICTION)
            .author("Herman Melville")
            .published(Year.of(isoYear:1851))
            .description(
                    "The book is the sailor Ishmael's narrative of the obsessive quest of "
                    + "Ahab, captain of the whaling ship Pequod, for revenge on Moby Dick, "
                    + "the giant white sperm whale that on the ship's previous voyage bit "
                    + "off Ahab's leg at the knee."
            )
            .build();
```

*error would not be in actual implementation

Sources

https://blogs.oracle.com/javamagazine/post/exploring-joshua-blochs-builder-design-pattern-in-java

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