תבניות עיצוב

תבניות עיצוב הן פתרונות מוכחים לבעיות נפוצות בתכנון תוכנה. הן לא פתרונות קוד מוכנים מראש, אלא תיאורים כללים של פתרונות אלו.

המטרות של תבניות עיצוב הן:

- להפוך את הקוד לקריא וברור יותר: שימוש בתבניות עיצוב מאפשר למפתחים לתקשר בצורה יעילה יותר, כי הן מספקות שפה משותפת לפתרונות עיצוביים.
- להפוך קוד לגמיש וניתן לשינוי: תבניות עיצוב מקלות עלינו לשנות את הקוד בקלות ובצורה יעילה. שינוי קל במקום אחד לא ישפיע על כל הקוד.

סוגי תבניות עיצוב

ישנן 3 קטגוריות של תבניות עיצוב:

1. תבניות יצירה (creational patterns): תבניות אלו מתמקדות ביצירת אובייקטים. הן מספקות דרך ליצור אובייקטים מסוגים שונים, כגון יצירת אובייקטים חדשים, יצירת אובייקטים מסוגים קיימים ועוד. תבניות מוכרות: Factory, Builder, Singleton.
2. תבניות מבנה (structural patterns):עוסקות באופן שבו אובייקטים מורכבים יחד למבנים גדולים ומורכבים יותר. תבניות מוכרות: Adapter, Decorator, Composite, Facade.
3. תבניות התנהגות (behavioral patterns): עוסקות באופן שבו אובייקטים מתקשרים זה עם זה. תבניות מוכרות: Observer, Strategy, Iterator.

חשוב לציין כי שימוש בתבניות עיצוב אינו תמיד הפתרון הטוב ביותר. חשוב להבין את הבעיה אותה אתם מנסים לפתור ולבחור את תבנית העיצוב המתאימה ביותר, תוך התחשבות ביתרונות ובחסרונות של כל תבנית.

Delegate

טכניקה ה Delegate היא טכניקה (ולא תבנית עיצוב) שמאפשרת לאובייקט להעביר את הפעולות שלו לאובייקט אחר. תבנית זו מאפשרת לנו להפריד בין הפעולות של האובייקט ואת המימוש שלהן.

דלגציה נותנת תחליף לירושה רגילה. במקום להשתמש בירושה כדי להעביר פעולות מאובייקט אחד לאחר, נשתמש בדלגציה. דלגציה מאפשרת לנו להעביר פעולות מאובייקט אחד לאחר בצורה יותר גמישה ופחות קשורה למחלקות.

היתרונות של Delegate הם:

- גמישות: דלגציה מאפשרת לנו להעביר פעולות מאובייקט אחד לאחר בצורה גמישה יותר.
- הפרדת אחריות: דלגציה מאפשרת לנו להפריד בין הפעולות של האובייקט ואת המימוש שלהן.
- שימוש חוזר בקוד: דלגציה מאפשרת לנו להשתמש בקוד שכתבנו בקלות ובפשטות.

דוגמא לשימוש ב Delegate

נניח שיש לנו מחלקה בשם Printer שאחראית על הדפסת מסמכים. במקום שהמחלקה תטפל בכל סוגי הקבצים שיש, ניצור מחלקה נוספת שתטפל בכל סוגי הקבצים ונעביר לה את הפעולה של ההדפסה.

```
class PDFPrinter{
    void printPDF(String file){
        // print the PDF file
    }
}

class WordPrinter{
    void printWord(String file){
        // print the Word file
    }
}

class Printer{
    void print(String file){
        if(file.endsWith(".pdf")){
```

```
            PDFPrinter pdfPrinter = new PDFPrinter();
            pdfPrinter.printPDF(file);
        } else if(file.endsWith(".doc")){
            WordPrinter wordPrinter = new WordPrinter();
            wordPrinter.printWord(file);
        }
    }
}
```

קוד ללא דלגציה יכול להיראות כך:

כאן המחלקה Printer יודעת להדפיס רק קבצי PDF ו-Word. אם נרצה להוסיף עוד סוג של קובץ נצטרך לשנות את הקוד של המחלקה Printer.

```
class Printer{
    void printPDF(String file){
        // print the PDF file
    }

    void printWord(String file){
        // print the Word file
    }

    void print(String file){
        if(file.endsWith(".pdf")){
            printPDF(file);
        } else if(file.endsWith(".doc")){
            printWord(file);
        }
    }
}
```

ההבדל בין דלגציה לירושה

ירושה היא תהליך בו מחלקה אחת מורישה את התכונות והפעולות של מחלקה אחרת. דלגציה היא תהליך בו מחלקה אחת מכירה את הפעולות של מחלקה אחרת ולוקחת אליה חלק מהפעולות שלה.

נתתמש בDelegation בתבנית העיצוב Strategy, ועוד.