

Inżynieria oprogramowania

LAB 5 Wzorce projektowe. Wzorzec Prototyp

PROTOTYP

Prototyp - w inżynierii oprogramowania jeden z obiektowych, konstrukcyjnych (kreacyjnych) wzorców projektowych, którego celem jest umożliwienie tworzenia obiektów danej klasy bądź klas z wykorzystaniem już istniejącego obiektu, zwanego prototypem. Głównym celem tego wzorca jest niezależenie systemu od sposobu w jaki tworzone są w nim produkty.

Omawiany wzorzec stosujemy między innymi wtedy, gdy nie chcemy tworzyć w budowanej aplikacji podklas obiektu budującego. Wzorzec ten stosujemy podczas stosowania klas specyfikowanych podczas działania aplikacji.

Wzorzec Prototyp powinien być używany, gdy:

- system powinien być niezależny od tego, jak jego produkty są tworzone, składane i reprezentowane;
- klasy, których egzemplarze należy tworzyć są specyfikowane w czasie wykonywania programu, np. przez dynamiczne ładowanie
- istnieje potrzeba uniknięcia budowania hierarchii klas fabryk, która jest porównywalna z hierarchią klas produktów
- stan obiektów klasy może przyjmować tylko jedną z kilku różnych wartości; może być wówczas wygodniej zainstalować odpowiednią liczbę prototypów i klonować je niż ręcznie tworzyć egzemplarze klasy za każdym razem z odpowiednim stanem.

Wzorzec prototypu określa rodzaj obiektów do tworzenia za pomocą prototypowej instancji. Prototypy nowych produktów są często budowane przed pełną produkcją, ale w poniższym przykładzie, prototyp jest bierny i nie bierze udziału w kopiowaniu siebie samego. Przykładem aktywnego prototypu (czyli biorącego udział w kopiowaniu siebie samego) jest biologiczny podział jednej komórki w dwie identyczne. Wtedy mamy do czynienia z klonowaniem.

PRZYKŁAD:

```
1 package wzorceProjektowePrototyp;
2
3 public class Cookie implements Cloneable {
4
5     private String Name;
6
7     public String getName() {
8         return Name;
9     }
10
11    public void setName(String name) {
12        this.Name = name;
13    }
14
15    public Object clone() {
16        try {
17            Cookie copy = (Cookie) super.clone();
18            return copy;
19        } catch (CloneNotSupportedException e) {
20            e.printStackTrace();
21            return null;
22        }
23    }
24 }
25
```

```
1 package wzorceProjektowePrototyp;
2
3 public class CoconutCookie extends Cookie{
4
5 }
```

```

1 package wzorceProjektowePrototyp;
2
3 public class CookieMachine {
4     private Cookie cookie;
5
6     public CookieMachine(Cookie cookie) {
7         this.cookie = cookie;
8     }
9
10
11    public Cookie makeCookie() {
12        return (Cookie)cookie.clone();
13    }
14 }

```

```

1 package wzorceProjektowePrototyp;
2
3 import java.util.ArrayList;
4
5 public class Prototyp {
6
7     public static void main(String[] args) {
8
9         ArrayList<Cookie> cookies = new <Cookie>ArrayList();
10
11        Cookie tempCookie = null;
12        Cookie prototyp = new CoconutCookie();
13        CookieMachine cm = new CookieMachine(prototyp);
14
15        for (int i = 1; i <= 100; i++) {
16            tempCookie = cm.makeCookie();
17            tempCookie.setName("CoconutCookie " + i);
18            cookies.add(tempCookie);
19        }
20
21        for (Cookie cookie : cookies) {
22            System.out.println(cookie.getName());
23        }
24    }
25 }

```

ZAD.1.

Utwórz projekt w NetBeans, który zawierał będzie klasy i metody zawarte w powyższym przykładzie. Przetestuj działanie tego programu.

ZAD.2.

Wzorując się na przedstawionym przykładzie utwórz klasę **Kartka** implementującą interfejs *Clonable*:

- Utwórz wewnątrz tej klasy prywatną zmienną tekstową **Text**
- Dodaj metody pozwalające na nadanie wartości tej zmiennej (**setText()**) oraz pobranie wartości z tej zmiennej (**getText()**)
- Dodaj metodę **kseruj()**, zwracającą klon klasy prototypu

Utwórz klasę **ProdukcjaKartek**:

- Utwórz wewnątrz tej klasy prywatną zmienną typu **Kartka**
- Utwórz konstruktor tej klasy, który będzie pobierał obiekt klasy **Kartka** i przypisywał jego wartość do zadeklarowanej wcześniej zmiennej prywatnej
- Utwórz metodę **zrobKsero()** zwracającą klon utworzony na zmiennej prywatnej

Utwórz klasę **Prototyp** i przetestuj działanie utworzonych klas i metod analogicznie jak w powyższym przykładzie