L04 Wzorce projektowe 1

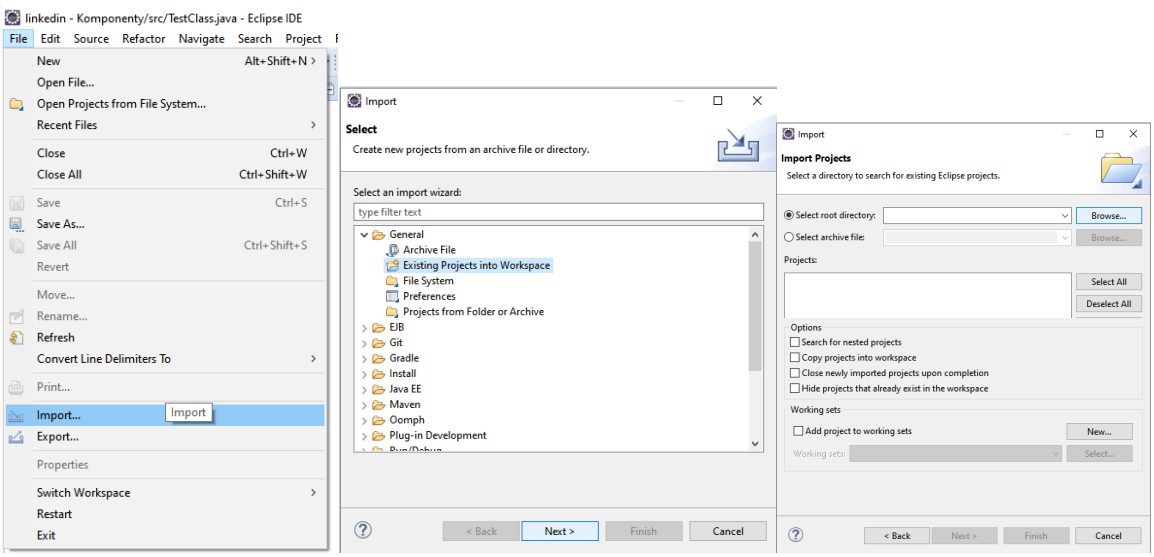
Ćwiczenia laboratoryjne

Cel

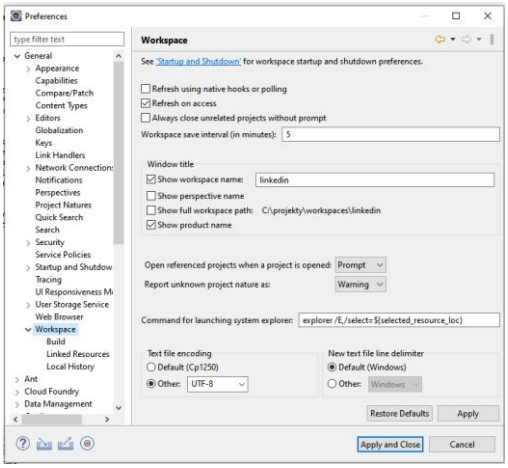
Celem ćwiczeń laboratoryjnych jest nabycie umiejętności korzystania ze wzorców projektowych w implementowaniu w języku java prostych przykładów projektów. Ćwiczenia są inspirowane książką „Rusz głową! Wzorce projektowe” wyd. Helion 2011.

Przygotowanie

Ze strony kursu proszę pobrać spakowany plik z kodem startowym, rozpakować do wybranego katalogu i zaimportować z tego katalogu do Eclipse istniejące projekty: SymulatorKaczki , Pogodynka.



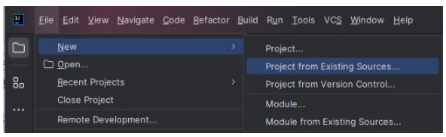
W treści zadań pozostawiono polskie znaki w nazwach klas itp., żeby wygodniej można było posługiwać się tym tekstem. Kod początkowy nie zawiera jednak polskich znaków. Rekomendowane jest, żeby rozwiązanie również nie zawierało polskich znaków.



Gdyby w miejscu polskich znaków pojawiły się krzaki, należy zmienić kodowanie polskich znaków w eclipse. W menu Window opcja Preferences i następnie > General, > Workspace i Text file encoding: UTF-8.

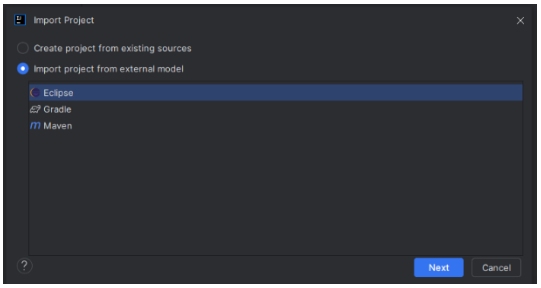
Środowisko IntelliJ

W IntelliJ uruchamianym po raz pierwszy należy utworzyć projekt bez nazwy aby uzyskać dostęp do pełnego menu. Następnie w menu wybrać *File/New/Project from Existing Sources…*.

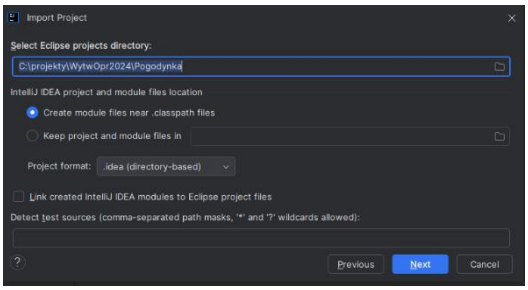


Alternatywnie można od razu, jako pierwszy krok, wybrać tę opcję za pomocą skrótu: *Ctrl+Shift+A*.

Potem wybrać katalog z projektem.
Następnie w oknie dialogowym wybrać opcję *Import project from external model* i *Eclipse* jako źródło projektu.



Kolejnym krokiem jest zaimportowanie projektu wraz z ustawieniami.

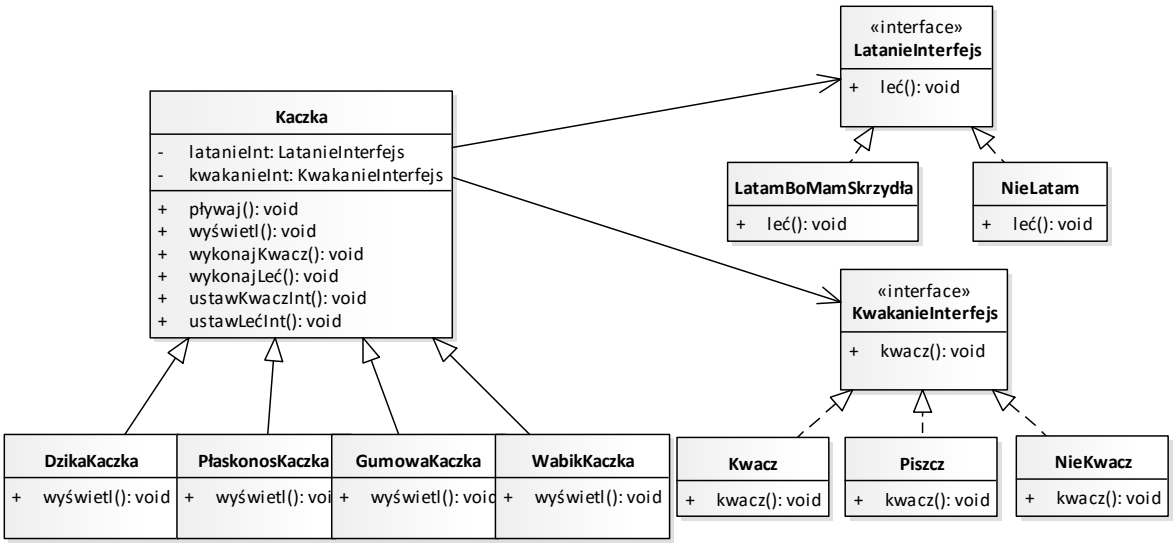


W kolejnych oknach dialogowych proszę naciskać *Next* i *Create.*

Sprawozdanie

Jako wynik pracy na zajęciach proszę przesłać pojedyncze archiwum w formacie „zip” zawierające projekty Eclipse’a/IntelliJ z rozwiązanymi zadaniami – czyli wystarczą pliki źródłowe.

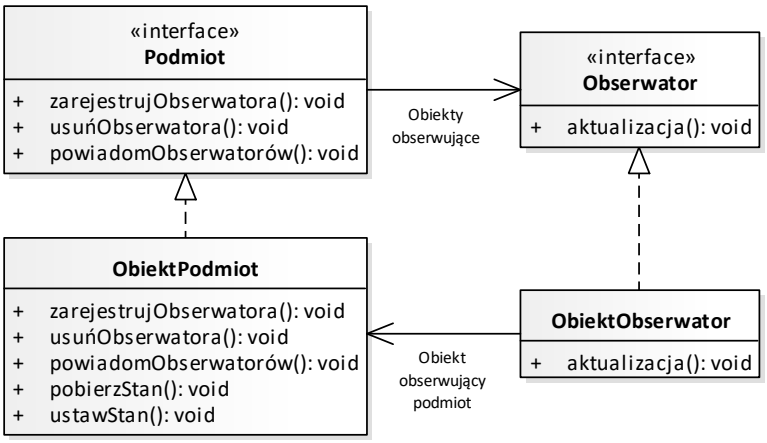
Zadanie 1 Symulator Kaczki



Proszę zaimplementować w projekcie SymulatorKaczki wzorzec projektowy Strategia w sposób omawiany na wykładzie, czyli:

1. W klasie Kaczka dodać zmienne obiektowe latanieInterfejs i kwakanieInterfejs, zdefinowane jako zmienne typu interfejs (odpowiedni) i usunąć z klasy Kaczka metody leć() i kwacz().
2. Dodać dwie podobne metody wykonajLeć() oraz wykonajKwacz(), które uruchamiają działanie ustawione w zmiennych latanieInterfejs i kwakanieInterfejs.
3. Przygotować klasy Kwacz, Piszcz, NieKwacz implementujące interfejs KwakanieInterfejs
4. Przygotować klasy LatamBoMamSkrzydła, NieLatam implementujące interfejs LatanieInterfejs.
5. Zaimplementować klasy DzikaKaczka, PłaskonosKaczka, GumowaKaczka, WabikKaczka używając odpowiednich implementacji interfejsów LatanieInterfejs i KwakanieInterfejs
6. Przetestować działanie kilku rodzajów kaczek za pomocą klasy testowej MiniSymulatorKaczki.
7. Utworzyć model kaczki z napędem rakietowym:
    a. Utworzyć nową klasę ModelKaczki, w której można dynamicznie, w trakcie działania programu zmieniać zachowania. Tylko obiekty tej klasy powinny mieć możliwość takiej dynamicznej zmiany – dla innych kaczek taka zmiana nie powinna być możliwa.
    b. Następnie utworzyć klasę LotZNapędemRakietowym implementującą odpowiedni interfejs.
    c. Przetestować dynamiczną zmianę sposobu latania kaczki w klasie testowej MiniSymulatorKaczki.

Zadanie 2 Pogodynka



Proszę zaimplementować w projekcie Pogodynka wzorzec projektowy Obserwator w sposób przedstawiony na wykładzie, czyli:

1. Utwórz klasy interfejsów `Podmiot, Obserwator i WyświetlElement`

```
public interface Podmiot {
    public void zarejestrujObserwatora(Obserwator o);
    public void usuńObserwatora(Obserwator o);
    public void powiadomObserwatorów();
}

public interface Obserwator {
    public void aktualizacja(float temperatura, float wilgotność,
    float ciśnienie);
}

public interface WyświetlElement {
    public void wyświetl();
}
```

2. Zmień klasę DanePogodowe:
    a. powinna implementować interfejs Podmiot
    b. powinna zawierać listę obserwatorów
    `private ArrayList<Obserwator> obserwatorzy = new ArrayList<>();`
    c. powinna implementować metody interfejsu Podmiot: zarejestrujObserwatora(), usuńObserwatora(), powiadomObserwatorów()
3. Zmień klasy wyświetlające dane pogodowe w następujący sposób:
    a. powinny implementować interfejs `Obserwator,`
    b. powinny implementować interfejs `WyświetlElement,`
    c. powinny obserwować dane pogodowe
4. Dostosuj klasę testującą StacjaMeteo i przetestuj jej działanie.

5. Dodaj nową klasę wyświetlającą indeks ciepła obliczony według wzoru
   indeksCiepła = 16.923 + 1.85212 * 10-1 * T + 5.37941 * RH - 1.00254 * 10-1 * T * RH + 9.41695 * 10-3 * T2 + 7.28898 * 10-3 * RH2 + 3.45372 * 10-4 * T2 * RH - 8.14971 * 10-4 * T * RH2 + 1.02102 * 10-5 * T2 * RH2 - 3.8646 * 10-5 * T3 + 2.91583 * 10-5 * RH3 + 1.42721 * 10-6 * T3 * RH + 1.97483 * 10-7 * T * RH3 - 2.18429 * 10-8 * T3 * RH2 + 8.43296 * 10-10 * T2 * RH3 - 4.81975 * 10-11 * T3 * RH3
   Gdzie:
   T -> temperatura,
   RH -> wilgotność,
   T2 = T^2
   T3 = T^3
   RH2 = RH^2
   RH3 = RH^3

```
private float computeHeatIndex(float t, float rh) {
        float index = (float)((16.923 + (0.185212 * t) + (5.37941 * rh) - (0.100254 * t * rh)
                        + (0.00941695 * (t * t)) + (0.00728898 * (rh * rh))
                        + (0.000345372 * (t * t * rh)) - (0.000814971 * (t * rh * rh)) +
                        (0.0000102102 * (t * t * rh * rh)) - (0.000038646 * (t * t * t)) + (0.0000291583 *
                        (rh * rh * rh)) + (0.00000142721 * (t * t * t * rh)) +
                        (0.000000197483 * (t * rh * rh * rh)) - (0.0000000218429 * (t * t * t * rh * rh)) +
                        0.000000000843296 * (t * t * rh * rh * rh)) -
                        (0.0000000000481975 * (t * t * t * rh * rh * rh)));
        return index;
}
```