Java – pakiety

Pakiety w Javie to mechanizm grupowania powiązanych ze sobą klas. Najłatwiej wyobrazić je sobie jako foldery (katalogach) na dysku komputera, w których segregujemy nasze pliki z kodem. **Deklarację pakietu umieszczamy zawsze na samej górze pliku z kodem**, np.: `package clinic.core;`.

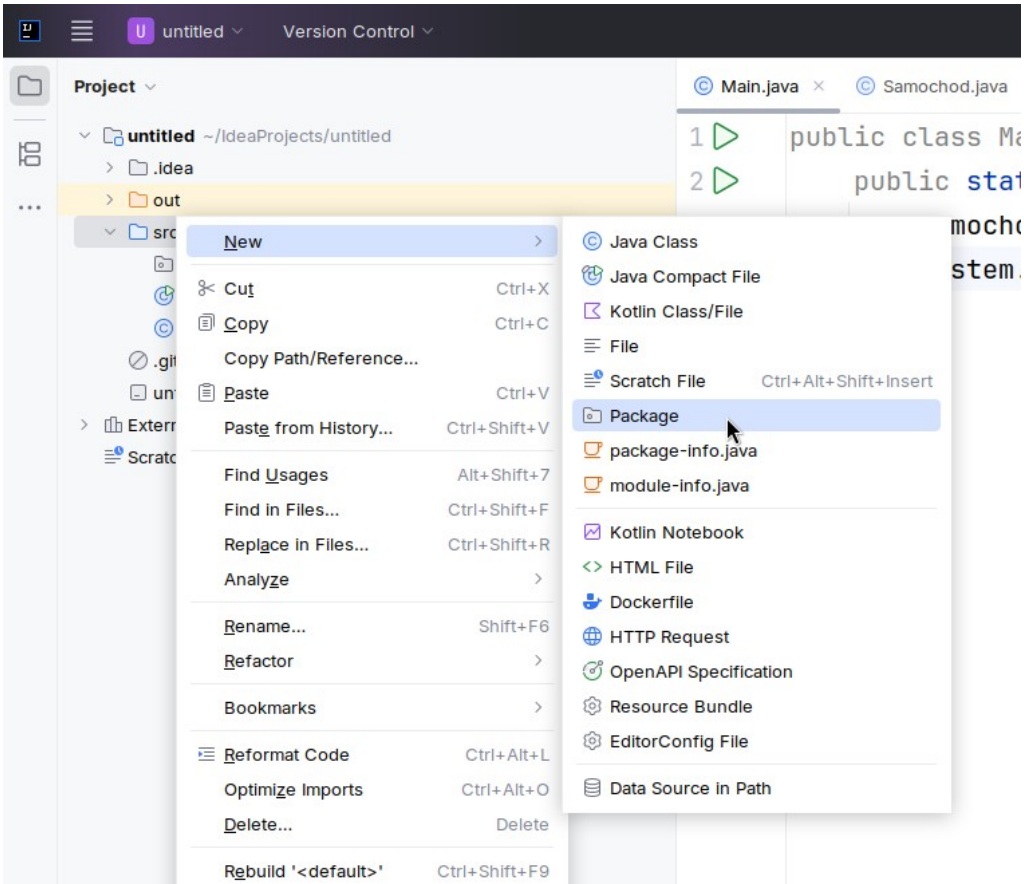
Każdy plik/klasa zdefiniowana w danym pakiecie powinna mieć zadeklarowany pakiet na samej górze przykładem powyżej.

Dlaczego używamy pakietów?

1. **Utrzymanie porządku (Organizacja)**: W dużych aplikacjach programiści tworzą setki, a nawet tysiące klas. Pakiety pozwalają pogrupować je tematycznie (np. osobny pakiet do łączenia z bazą danych, osobny do logiki biznesowej, a osobny do wyświetlania okienek użytkownikowi).
2. **Unikanie konfliktów nazw**: Dzięki pakietom możemy mieć w jednym projekcie dwie klasy o dokładnie tej samej nazwie, o ile znajdują się w różnych pakietach (np. klasa `Date` istnieje w Javie zarówno w pakiecie `java.util`, jak i `java.sql`).
3. **Kontrola dostępu:** Pakiety odgrywają ogromną rolę w bezpieczeństwie i hermetyzacji kodu. Jeśli przed zmienną lub metodą nie postawimy żadnego modyfikatora (czyli nie napiszemy ani `public`, ani `private`), Java zastosuje tzw. dostęp domyślny (`package-private`). Oznacza to, że dana metoda lub zmienna będzie widoczna tylko i wyłącznie dla innych klas **znajdujących się w tym samym pakiecie**. Klasy z innych pakietów w ogóle nie będą miały do niej dostępu.

Tworzenie pakietu w IntelliJ IDEA

Aby utworzyć pakiet w środowisku IntelliJ IDEA należy z drzewa struktury projektu (lewy panel z folderami i plikami) na folderze `src` nacisnąć Prawym Przyciskiem Myszy, a następnie wybrać z menu kontekstowego opcję `New` > `Package`.



Zadanie - System Rejestracji Pacjentów

Zadanie polega na stworzeniu prostej aplikacji symulującej system w przychodni. Celem jest poprawne ukrycie danych wrażliwych i udostępnienie tylko tych metod, które są bezpieczne dla użytkownika zewnętrznego, przy jednoczesnym pozwoleniu na szerszy dostęp klasom wewnątrz systemu przychodni.

1. Zacznijmy od utworzenia nowego projektu oraz dodaniu dwóch pakietów o nazwach `clinic.core` oraz `clinic.client`.

Część 1: Pakiet `clinic.core` (Logika wewnętrzna przychodni)

1. Utwórz klasę `Pacjent` wewnątrz pakietu `clinic.core`.

   A) Klasa powinna posiadać następujące pola: `name` (String), `pesel` (String), `kodChoroby` (String).

   B) **Wymóg:** Zabezpiecz pola tak, aby nie dało się ich w żaden sposób odczytać ani zmienić bezpośrednio z innych klas (pełna hermetyzacja).

   C) Utwórz konstruktor, który przyjmuje i ustawia `name` oraz `pesel`. Pole `kodChoroby` powinno być na start puste (np. `""`).

   D) Utwórz publiczną metodę `getName()`, która zwraca imię pacjenta.

   E) Utwórz metodę setKodChoroby`(String code)`, która pozwala na zmianę kodu choroby, ale **tylko** klasom znajdującym się w tym samym pakiecie (np. lekarzom wewnątrz przychodni). Metoda ta **nie** może być publiczna.

2. Utwórz klasę `Doktor` wewnątrz pakietu `clinic.core`.

   A) Klasa posiada publiczną metodę `diagnoza(Pacjent pacjent, String kod)`.

   B) Wewnątrz tej metody lekarz powinien przypisać nowy kod choroby do pacjenta (korzystając z metody pacjenta zdefiniowanej wyżej).

   Klasa `Doktor` nie będzie miała żadnych pól oraz kontruktora, jedynie będzie posiadać metodę z punktu 2A. Zauważ że w metodzie `diagnoza` jest podawany jako parametr obiekt pacjenta. Dzięki temu z tej klasy będziemy mogli wywoływać metody obiektu `Pacjent`.

Część 2: Pakiet `clinic.client` (Aplikacja zewnętrzna - np. recepcja/pacjent)

1. Utwórz klasę `ReceptionApp` z metodą `main`. Przypominam, że metoda `main` musi być publiczna, statyczna oraz zawierać parametr `String[] args`!

2. W metodzie `main`:

   - Utwórz obiekt klasy `Pacjent` oraz obiekt klasy `Doktor`. Zwróć uwagę na to, że klasa `Doktor` nie zawiera konstruktora, bo go nie napisaliśmy. Dalej możemy stworzyć obiekt tej klasy ponieważ Java sama dodaje pusty, bezargumentowy kontruktor. Dlatego podczas nowego obiektu `Doktor` nie podajemy żadnych argumentów dla konstruktora.
   - Wypisz na konsoli imię pacjenta.
   - Spróbuj odwołać się do pola `pesel` pacjenta. Zakomentuj tę linijkę kodu i w komentarzu obok napisz jednym zdaniem, **dlaczego kompilator zgłasza błąd**.
   - Spróbuj wywołać metodę `setKodChoroby(...)` bezpośrednio na obiekcie pacjenta z poziomu klasy `ReceptionApp`. Zakomentuj tę linijkę i w komentarzu obok napisz, **dlaczego kompilator zgłasza błąd**.
   - Wywołaj poprawnie metodę `diagnoza(...)` na obiekcie lekarza, przekazując pacjenta i przykładowy kod choroby.

Zasady oceniania

Maksymalna ilość puntków do uzyskania: 10

Ocenie podlegać będą:

- struktura pakietów [2 pkt.],
- hermetyzacja w klasie Pacjent [2 pkt.],
- interfejs publiczny [2 pkt.],
- dostęp pakietowy [2 pkt.],
- zrozumienia ograniczeń (klasa ReceptionApp) [2 pkt.].