Hermetyzacja (z ang. *encapsulation*, **kapsułkowanie** lub inaczej **ukrywanie informacji**) . Polega ono na ukrywaniu pewnych danych składowych lub metod obiektów danej klasy tak, aby były one (i ich modyfikacja) dostępne tylko metodom wewnętrznym danej klasy. Z **pełną hermetyzacją** mamy do czynienia wtedy, gdy dostęp do wszystkich pól w klasie jest możliwy tylko i wyłącznie poprzez metody, lub inaczej mówiąc gdy wszystkie pola w klasie znajdują się w sekcji prywatnej (lub chronionej).

Ćwiczenie 1

Napisz program, który:

- **utworzy klasę Osoby** zawierającą prywatne pola typu string : **PESEL, nazwisko, imie, data_urodzenia,** oraz pole typu **unsigned int wiek** .

Ćwiczenie 2

Napisz program, który:

- **utworzy klasę Osoby** zawierającą prywatne pola typu string : **PESEL, nazwisko, imie, data_urodzenia,** oraz pole typu **unsigned int wiek** oraz
- konstruktor klasy **Osoby**, inicjujący wartości obiektu klasy Osoby: **Osoby(string p, string n, string i, string d, unsigned int w) ,**

Ćwiczenie 3

Napisz program, który:

- **utworzy klasę Osoby** zawierającą prywatne pola typu string : **PESEL, nazwisko, imie, data_urodzenia,** oraz pole typu **unsigned int wiek** oraz
- konstruktor klasy **Osoby**, inicjujący wartości obiektu klasy Osoby: **Osoby(string p, string n, string i, string d, unsigned int w) ,**
- zdefiniuje metody pobierające dla klasy **Osoby**, zwracające wartości pól prywatnych: **GetPesel, GetNazwisko, GetImie, GetDataUrodzenia, GetWiek**

Ćwiczenie 4

Napisz program, który:

- **utworzy klasę Osoby** zawierającą prywatne pola typu string : **PESEL, nazwisko, imie, data_urodzenia,** oraz pole typu **unsigned int wiek** oraz
- konstruktor klasy **Osoby**, inicjujący wartości obiektu klasy Osoby: **Osoby(string p, string n, string i, string d, unsigned int w) ,**
- zdefiniuje metody pobierające dla klasy **Osoby**, zwracające wartości pól prywatnych: **GetPesel, GetNazwisko, GetImie, GetDataUrodzenia, GetWiek**
- zdefiniuje metody ustawiające dla klasy **Osoby**, ustawiające wartości pól prywatnych: **PutPesel, PutNazwisko, PutImie, PutDataUrodzenia, PutWiek**

Ćwiczenie 5

Napisz program, który:

- **utworzy klasę Osoby** zawierającą prywatne pola typu string : **PESEL, nazwisko, imie, data_urodzenia,** oraz pole typu **unsigned int wiek** oraz
- konstruktor klasy **Osoby**, inicjujący wartości obiektu klasy Osoby: **Osoby(string p, string n, string i, string d, unsigned int w) ,**
- zdefiniuje metody pobierające dla klasy **Osoby**, zwracające wartości pól prywatnych: **GetPesel, GetNazwisko, GetImie, GetDataUrodzenia, GetWiek**
- zdefiniuje metody ustawiające dla klasy **Osoby**, ustawiające wartości pól prywatnych: **PutPesel, PutNazwisko, PutImie, PutDataUrodzenia, PutWiek**
- Zainicjuj obiekt osoba klasy **Osoby** i wyświetl **PESEL, nazwisko, imię, datę urodzenia, wiek** dla tego obiektu,
- Napisz metodę sprawdzającą poprawność numeru PESEL w oparciu o poniższy algorytm oraz **sprawdź numer PESEL dla przykładowego obiektu klasy Osoby**.

Liczba kontrolna i sprawdzanie poprawności numeru

Jedenasta cyfra jest cyfrą kontrolną, służącą do wychwytywania przekłamań numeru. Jest ona generowana na podstawie pierwszych dziesięciu cyfr. Aby sprawdzić czy dany PESEL jest prawidłowy należy, zakładając, że litery *a-k* to kolejne cyfry numeru od lewej, obliczyć wyrażenie

```
a + 3*b + 7*c + 9*d + e + 3*f + 7*g + 9*h + i + 3*j + k
```

Przykładowy wynik:

```
76032205555 Kowalski Jan 760322   32
Numer PESEL OK
```