1. Stwórz klasę Osoba z polami imię i nazwisko. Następnie stwórz klasę potomną Student z polami rokStududiów, numerGrupy, numerAlbumu. Dodaj w obu klasach konstruktory domyślne i parametryczne (można wykorzystać kod z ćw nr 4).
2. W obu klasach stwórz metody **`WypiszInfo()`** wyświetlające wszystkie pola z klasy na konsoli (mamą to być tylko instrukcje **`Console.WriteLine()`** z parametrami).
3. Dodaj **`new`** aby pozbyć się ostrzeżenia (warning) w Visual Studio. Czemu takie rozwiązanie nie jest najlepsze?
4. Wykonaj rzutowanie w górę (**`Osoba student1 = new Student()`**) i wywołaj metodę **`WypiszInfo()`**.
5. Usuń **`new`** (dodane w punkcie 3). Następnie do metody **`WypiszInfo()`** w klasie Osoba dodaj **`virtual`**, a w klasie Student **`override`**. Jaka jest różnica?
6. Zmodyfikuj metodę **`WypiszInfo()`** w klasie Student używając **`base.WypiszInfo();`** .
7. Stwórz nowy projekt. Dodaj abstrakcyjną klasę Figura, a w niej pola a,b,c (z modyfikatorem protected) i abstrakcyjną metodą ObliczPole().
8. Stwórz dwie klasy potomne Kwadrat i Trójkąt dziedziczące z klasy Figura.
9. Stwórz obiekty z klasy Figura, Kwadrat, Trójkąt (w razie potrzeby stwórz konstruktory). Czy wszystko jest możliwe?
10. Dodaj w klasach Kwadrat i Trojkat przesłoniętą metodę ObliczPole() i wywołaj ją dla stworzonych obiektów.
11. Stwórz listę obiektów z klas pochodnych do klasy Figura. Następnie za pomocą instrukcji foreach oblicz dla nich pola.
12. Odszukaj kod z ćwiczeń nr 4, gdzie należało stworzyć grę.
    a. stwórz klasę na przechowywanie drużyny bohaterów (domyślnie 10 osób), drużyny mogą przechowywać bohaterów różnych typów (pamiętaj o właściwym rzutowaniu).
    b. dodaj metodę kopiująca bohaterów korzystającą z interfejsu IClonable.
    c. dodaj metodę na sortowanie bohaterów o żywotności lub liczbie punktów taktyki
    d. klasę rodzica np. Bohater przerób na klasę abstrakcyjną, ogranicz niepotrzebne dublowanie kodu,
    e. przetestuj działanie na kilku drużynach.