PRZYKŁAD 1. Na podstawie poniższego kodu odpowiedz, dlaczego klasa implementuje kilka interfejsów zamiast jednego "dużego"?

```
public interface WydajeDzwiek{
        public void graj();
}
public interface WyswietlaObraz{
        public void wyswietl();
}
public class Radio implements WydajeDzwiek{
        public void graj(){
                System.out.println("gra");
        }
}
public class Telewizor implements WydajeDzwiek, WyswietlaObraz{
        public void graj(){
                System.out.println("gra");
        }
        public void wyswietl(){
                System.out.println("wyswietla");
        }
}
```

PRZYKŁAD 2. Utwórz klasę, która implementuje oba interfejsy:

```
public interface A{
        public void f ();
}
```

```
public interface B{
        public void f ();
}
```

PRZYKŁAD 3. Utwórz klasę, która implementuje oba interfejsy:

```
public interface A{
        public void f (int liczba);
}
```

```
public interface B{
        public int f (float liczba);
}
```

PRZYKŁAD 4. Utwórz klasę, która implementuje oba interfejsy:

```
public interface A{
        public void f ();
}
```

```
public interface B{
        public int f ();
}
```

PRZYKŁAD 5. Utwórz klasę, która implementuje interfejs B:

```
public interface A{
        public void f ();
}
```

```
public interface B extends A{
        public void g ();
}
```

PRZYKŁAD 6. Utwórz klasę, która implementuje interfejs C:

```
public interface A{                    public interface B{
        public void f();                       public void g();
}
                              }
public interface C extends A, B{
        public void h();
}
```

PRZYKŁAD 7. Oceń poprawność dziedziczenia interfejsów:

```
public interface A{                    public interface B{
        public int f ();                       public float f ();
}                                      }

public interface C extends A, B{
        public void h();
}
```

Zadanie 12

Dla każdego naukowca i studenta (zad.7) chcemy mieć przyporządkowany zbiór liczb ważnych dla danej osoby (numery telefonów, kart bibliotecznych, numery PIN).

1. Klasę **Osoba** przekształć w interfejs **OsobaInterfejs.**
   - co się stanie z polami klasy **Osoba**?
   - co zmieni się w jej metodach?
2. Zamiast klasy **Naukowiec** zaprojektuj klasę **NaukowiecZbior**, będącą klasą pochodną od klasy **ZbiorLiczb** i implementującą interfejs **OsobaInterfejs.** Podobnie dla klasy **StudentZbior.**
3. Referencje do obiektów n1, n2 typu **NaukowiecZbior** i s1, s2 typu **StudentZbior** można wpisać do tablicy **tab** typu **OsobaInterfejs**. Które metody zachodzą w sposób polimorficzny?
4. Nie każde wywołania mogą być polimorficzne. Jaki będzie wynik działania poniższego i dlaczego?

   tab[i].Wszystkie();

   Wskazówka: Aby dla elementu tablicy **tab[i]** wywołać jakąś metodę z klasy **ZbiorLiczb** trzeba najpierw sprawdzić, jakiego typu jest referencja **tab[i]** w następujący sposób:

   tab[i] ***instanceof*** *NaukowiecZbior*

   Wynikiem jest **true/false**. Następnie dokonujemy jawnej konwersji, np:

   liczby = ((NaukowiecZbior) tab[i]).Wszystkie();