[GUI] Ćwiczenia I:

```
public class Test {
    public static void main(String[] args) {
        Muzyk[] muzycy = {new Skrzypek("Aleks", 2) // Imie, czas wystapienia (w
godz.)
        new Wiolonczelista("Bartek", 1),
        new Flecista("Czarek", 0.5)};
        for (Muzyk m : muzycy)
            System.out.println("Muzyk: " + m.imie() + '\n' +
                "Instrument: " + m.instrument() + '\n' +
                "Czas wystąpienia: " + m.czas() + " godz. " + '\n' +
                "Stawka godzinowa: " + m.stawka() + '\n');

        System.out.println(Muzyk.maxHonorarium(muzycy));        // muzyk otrzymu
jący najwyższe honorarium za występ
    }
}
```

```
abstract class Muzyk {

    private String imie;
    private double czas;

    // konstruktor
    protected Muzyk(...) {
        //...
    }

    // metoda getter
    protected String imie() {
        //...
    }

    // metoda getter
    protected double czas() {
        //...
    }

    // metody abstrakcyjne
    abstract protected String instrument();
    abstract protected double stawka();

    public String toString() {
        //...
    }

    public static Muzyk maxHonorarium(Muzyk[] muzycy)
    {
```

```
        //...
    }
}
```

```
// podklasa dziedzicząca po klasie abstrakcyjnej
class Flecista extends Muzyk {
    // konstruktor
    public Flecista(...) {
        //...
    }

    @Override
    public String instrument() {
        return "Flet";
    }

    @Override
    public double stawka() {
        return 300.0;
    }
}
```

```
interface In1 {
    // publiczne metody abstrakcyjne
    public void metoda1();
    public String method1();

    // publiczna statyczna stała
    public static final int STALA = 1;
}

interface In2 {
    // (publiczne) metody abstrakcyjne
    void metoda2();
    String method2();

    // (publiczna) metoda domyślna
    default void methDef2() { System.out.println("Default method"); }

    // (publiczna) statyczna metoda
    static void meth2() {}
}
```

```
class K implements In1, In2 {

    @Override
    public void metoda1() { System.out.println("Metoda1");}

    @Override
```

```
    public String method1() { return "method1"; }

    @Override
    public void metoda2() { System.out.println("Metoda2");}

    @Override
    public String method2() { return "method2";}

//  @Override
//  public void methDef2() { System.out.println("Default method 2");}
}
```

```
class K2 extends K {
    public void metoda() { System.out.println("Metoda");}
}
```

```
public class gui_note_03_01 {

    public static void main(String[] args) {

        In1 o = new K();
        o.metoda1();            // "Metoda1"
        //o.metoda2();          // Błąd

        In2 ob = new K();
        ob.metoda2();           // "Metoda2"
        ob.methDef2();          // "Default method"

        //K2 obi = new K();     // Błąd
        K obi = new K2();       // OK

        //obi.metoda();         // Błąd
        ((K2)obi).metoda(); // "Metoda

        System.out.println(ob instanceof In2);          // true

        System.out.println(ob instanceof K);            // true
        System.out.println(ob instanceof K2);           // false

        System.out.println(obi instanceof K);           // true
        System.out.println(obi instanceof K2);          // true

        System.out.println(ob.getClass().getName());    // "K"
        System.out.println(obi.getClass().getName());   // "K2"
    }
}
```