8. Odnośniki

[1] Klasa Pojazd i jej pochodne:

```
public abstract class Pojazd
{
    public string Marka { get; set; }
    public string TypSilnika { get; set; } // np. benzyna, diesel, elektryczny
    public int MaksymalnaPrędkość { get; set; }
    public double Spalanie { get; set; } // litry na 100 km
    public double Ciezar { get; set; } // w kg
    public decimal Cena { get; set; } // Cena pojazdu

    public abstract void WyswietlInformacje();
    public virtual decimal KosztPrzejazduNa100km(decimal cenaPaliwa)
    {
        return (decimal)Spalanie * cenaPaliwa;
    }
}
```

[2] Klasa Pracownik i jej pochodne:

```
public abstract class Pracownik
{
    public string Imie { get; set; }
    public string Nazwisko { get; set; }
    public string Stanowisko { get; set; }
    public decimal Wynagrodzenie { get; set; }

    public abstract void PokazInformacje();
    public virtual decimal ObliczDzienneWynagrodzenie()
    {
        return Wynagrodzenie / 22;
    }

    public virtual decimal ObliczMiesieczneWynagrodzenie()
    {
        return Wynagrodzenie;
    }
    public virtual decimal ObliczRoczneWynagrodzenie()
    {
        return Wynagrodzenie * 12;
    }
}
```

[3] Implementacja metod wynagrodzenia w klasie bazowej Pracownik:

```
public virtual decimal ObliczDzienneWynagrodzenie()
{
    return Wynagrodzenie / 22;
}

public virtual decimal ObliczMiesieczneWynagrodzenie()
{
    return Wynagrodzenie;
}

public virtual decimal ObliczRoczneWynagrodzenie()
{
    return Wynagrodzenie * 12;
}
```

[4] Implementacja metod wynagrodzenia dla klasy PracownikBiurowy:

```
public override decimal ObliczDzienneWynagrodzenie()
{
    decimal stawkaGodzinowa = Wynagrodzenie / IloscGodzinPracy;
    return stawkaGodzinowa * 8;
}

public override decimal ObliczMiesieczneWynagrodzenie()
{
    decimal wynagrodzeniePodstawowe = Wynagrodzenie;
    decimal bonusZaNadgodziny = (IloscGodzinPracy > 160) ? (IloscGodzinPracy - 160) * (Wynagrodz
    return wynagrodzeniePodstawowe + bonusZaNadgodziny;
}

public override decimal ObliczRoczneWynagrodzenie()
{
    return ObliczMiesieczneWynagrodzenie() * 12;
}
```

[5] Implementacja metod wynagrodzenia dla klasy Menedzer:

```
public override decimal ObliczDzienneWynagrodzenie()
{
    return Wynagrodzenie / 22;
}

public override decimal ObliczMiesieczneWynagrodzenie()
{
    return Wynagrodzenie;
}

public override decimal ObliczRoczneWynagrodzenie()
{
    return ObliczMiesieczneWynagrodzenie() * 12 + BonusRoczny;
}
```

[6] Deklaracja klasy PracownikZdalny z właściwością Terminowosc:

```
public class PracownikZdalny : Pracownik
{
    public int IloscDniZdalnych { get; set; }
    public bool Terminowosc { get; set; }

    // Metody wyświetlania i obliczania wynagrodzeń...
}
```

[7] Implementacja metod wynagrodzenia dla klasy PracownikZdalny:

```
public override decimal ObliczDzienneWynagrodzenie()
{
    decimal podstawowaDniowka = Wynagrodzenie / IloscDniZdalnych;
    return Terminowosc ? podstawowaDniowka * 1.1m : podstawowaDniowka; //bonus 10%
}

public override decimal ObliczMiesieczneWynagrodzenie()
{
    decimal podstawoweWynagrodzenie = Wynagrodzenie;
    return Terminowosc ? podstawoweWynagrodzenie * 1.15m : podstawoweWynagrodzenie; //bonus 15%
}

public override decimal ObliczRoczneWynagrodzenie()
{
    decimal podstawoweRoczne = ObliczMiesieczneWynagrodzenie() * 12;
    return Terminowosc ? podstawoweRoczne + Wynagrodzenie : podstawoweRoczne;
}
```