Dziedziczenie

Dana jest klasa Punkt w pliku o nazwie Punkt.java:

```
public class Punkt{
  int x, y;
  Punkt(){
    x = 1;
    y = 1;
  }
  Punkt(int wspX, int wspY) {
    x = wspX;
    y = wspY;
  }
  Punkt(Punkt A) {
    x = A.x;
    y = A.y;
  }
  int pobierzX(){
    return x;
  }
  int pobierzY(){
    return y;
  }
  void wyswietl() {
    System.out.println("x = " + x + ", y = " + y);
  }
}
```

Zad.17. (*patrz wykład*) Utwórz nowy projekt, umieść w nim plik Punkt.java z kodem klasy Punkt. W nowym pliku utwórz klasę Punkt3D, służącą do przechowywania punktów w trzech wymiarach, dziedziczącą po klasie Punkt:

```
public class Punkt3D extends Punkt{
  int z;
}
```

Korzystając z metod klasy Punkt i składni `super`, dopisz do klasy Punkt3D:

1. konstruktor bezargumentowy,
2. konstruktor przyjmujący trzy argumenty typu int,
3. konstruktor przyjmujący parametr typu Punkt i parametr typu int,
4. konstruktor przyjmujący parametr typu Punkt3D,
5. metodę `void wyswietl();` - wypisującą wartości trzech współrzędnych, przesłaniającą metodę o tej samej nazwie z klasy Punkt,
6. metodę zwracającą wartość pola z.

W nowym pliku utwórz klasę Main z metodą main i przetestuj wszystkie napisane metody.

Zad.18. W osobnych plikach utwórz klasy o nazwie

- Bazowa, zawierającą pole typu `int` o nazwie `liczba`,
- Potomna, dziedziczącą po klasie Bazowa i również zawierającą pole `liczba`.

W klasie Potomna zdefiniuj metodę `pobierzWartosc`, przyjmującą jeden argument typu `boolean`. Jeśli wartością argumentu będzie
- true: metoda zwróci wartość pola `liczba` zdefiniowanego w klasie Potomna,
- false: metoda zwróci wartość pola `liczba` odziedziczonego po klasie Bazowa.

W klasie Main i metodzie main utwórz obiekt klasy Potomna i wywołaj metodę `pobierzWartosc`.

Zad.19. Do klasy Potomna z poprzedniego zadania dopisz metodę `ustawWartosc` przyjmującą dwa argumenty: pierwszy typu `int`, drugi typu `boolean`. Jeśli wartością drugiego argumentu będzie
- true: wartość pierwszego przypisz polu `liczba` zdefiniowanemu w Potomna,
- false: wartość pierwszego przypisz polu `liczba` zdefiniowanemu w klasie Bazowa.

W klasie Main i metodzie main utwórz obiekt klasy Potomna i wywołaj metodę `ustawWartosc`.

Zad.20. Utwórz klasę Auto, która zawiera:

- pole `float[] przebieg;` - tablicę przechowującą informację o liczbie przejechanych kilometrów w kolejnych miesiącach, jej rozmiar to 12; tablica jest inicjalizowana dowolnymi wartościami w konstruktorze.
- metodę `float srPrzebieg();` - obliczającą średni przebieg dla samochodu.

Utwórz klasę Taxi, która dziedziczy po klasie Auto i zawiera:

- pole `float[] zarobki;` - tablicę przechowującą informację o zarobkach taksówkarza w kolejnych miesiącach, jej rozmiar to 12; tablica jest inicjalizowana dowolnymi wartościami w konstruktorze.
- metodę `float srZarobki();` - obliczającą średnie zarobki dla taksówki.

W klasie Main i metodzie main utwórz obiekt klasy Taxi i wyświetl na ekranie średni przebieg i średnie zarobki.

Polimorfizm

Zad.21. W którym miejscu poniższego kodu można zaobserwować polimorfizm? Na czym on polega?

```
public class Figura {
  float f, g;
  Figura(float a, float b) {
    f = a;
    g = b;
  }
  float pole() {
    System.out.println("Niezdefiniowane");
    return 0;
  }
}
```

```
public class Prostokat extends Figura {
  Prostokat(float a, float b) {
    super(a,b);
  }
  float pole() {
    System.out.println("Prostokat");
    return f*g;
  }
}
```

```
public class Trojkat extends Figura {
  Trojkat(float a, float b) {
    super(a,b);
  }
  float pole() {
    System.out.println("Trojkat");
    return f*g/2;
  }
}
```

```
public class Test {
  public static void main(String args[]){
    Figura fig  = new Figura(1,4);
    Prostokat p = new Prostokat(5,10);
    Trojkat t   = new Trojkat (6,2);
    Figura  a;
    a = p;    System.out.println(a.pole());
    a = t;    System.out.println(a.pole());
    a = fig;  System.out.println(a.pole());
  }
}
```

Zad.22. Przeanalizuj i przetestuj podany kod.

```
public class Telefon
{
  private String nrTel;
  private int lacznyCzas;
  private static double cenaRozmowy = 0.48;
  Telefon (String numer) {
    nrTel = numer;
  }
  double obliczKwoteDoZaplaty() {
    return cenaRozmowy * (lacznyCzas / 60);
  }
  static void ustawCeneRozmowy(double nowaCena){
    cenaRozmowy = nowaCena;
  }
  void zadzwon(String nr) {
    System.out.println("Dzwonie do: " + nr);
    System.out.println("Rozmowa w toku...");
    int czasRozmowy = (int)(Math.random()*3600);
    lacznyCzas = lacznyCzas + czasRozmowy;
    System.out.println("Rozmowa zakonczona.");
    System.out.print("Czas rozmowy: "+ czasRozmowy/60 + "min, "
                     + czasRozmowy%60 + "sek.");
  }
}
```

```
public class Main
{
  public static void main(String[] args){
    Telefon telAla = new Telefon("783982331");
    Telefon telOla = new Telefon("608234982");
    telAla.zadzwon("0124239832");
    telOla.zadzwon("112");
    double kwota = telAla.obliczKwoteDoZaplaty();
    System.out.println("Ala ma do zapłaty: " + kwota + " zl.");
  }
}
```

Z klasy Telefon wyprowadź dwie klasy pochodne: TelefonKomorkowy i TelefonStacjonarny. Każda z tych klas powinna posiadać

- nowe pola (np. operator, prefiks),
- nowe metody (np. wyślijSMS),
- własną implementację metody `zadzwon(String)` (przesłanianie).

Utwórz tablicę typu Telefon, np:

```
Telefon[] tablicaTelefonow = new Telefon[3];
tablicaTelefonow[0] = new Telefon("634295432");
tablicaTelefonow[1] = new TelefonKomorkowy("504295432", "T-mobile");
tablicaTelefonow[2] = new TelefonStacjonarny("126493042", "058");
```

Zauważ, że poszczególne elementy tablicy zawierają referencje zarówno do obiektów klasy Telefon, jak również do obiektów klas pochodnych. Dla każdego elementu tablicy wywołaj metodę `zadzwon`. Czy wywołania te były polimorficzne? Dlaczego?