Programowanie w Javie
Lista nr 2

1. Dziedziczenie

```
3  class TAdres
4  {
5      private String Panstwo;
6      private String Miasto;
7      private String Ulica;
8      public TAdres()
9      {
10
11     }
12     public TAdres(String n_p, String n_m, String n_u)
13     {
14         this.Panstwo=n_p;
15         this.Miasto=n_m;
16         this.Ulica=n_u;
17     }
18     public String dajPanstwo()
19     {
20         return Panstwo;
21     }
22     public String dajMiasto()
23     {
24         return Miasto;
25     }
26     public String dajUlice()
27     {
28         return Ulica;
29     }
30     public void Wypisz()
31     {
32         System.out.println("Państwo: "+dajPanstwo()+", Miasto: "+dajMiasto()+", Ulica :"+dajUlice());
33     }
34 }
```

- 3 – początek klasy podstawowej TAdres
- 8 – konstruktor bezparametrowy
- 12 – konstruktor parametrowy

```
36  final class  TAdresSzczegolowy extends TAdres
37  {
38      private int NrDomu;
39      private int NrMieszkania;
40      public TAdresSzczegolowy()
41      {
42      }
43      public TAdresSzczegolowy(String n_p, String n_m, String n_u, int n_nr, int n_nr_miesz)
44      {
45          super(n_p,n_m,n_u);
46          this.NrDomu=n_nr;
47          this.NrMieszkania=n_nr_miesz;
48      }
49      public int dajNrdomu()
50      {
51          return NrDomu;
52      }
53      public int dajNrMieszkania()
54      {
55          return NrMieszkania;
56      }
57      @Override
58      public void Wypisz()
59      {
60          super.Wypisz();
61          System.out.println("nr domu: "+dajNrdomu()+", nr mieszkania: "+dajNrMieszkania());
62      }
63  }
```

- 36 – klasa TAdresSzczegolowy dziedzicząca po klasie TAdres. Modyfikator **final** oznacza, że po tej klasie nie będzie już możliwości dziedziczenia
- 40 - konstruktor bezparametrowy
- 43 – konstruktor parametrowy.
- 45 – jawne wywołanie konstruktora klasy przodka.
- 60 – jawne wywołanie przeładowanej (@Override) metody przodka.

```
67  public static void main(String[] args) {
68      TAdres adres1 = new TAdres( n_p: "Polska",  n_m: "Nysa",  n_u: "Rynek");
69      adres1.Wypisz();
70      TAdresSzczegolowy adres2 = new TAdresSzczegolowy( n_p: "Polska",  n_m: "Nysa",  n_u: "Rynek",  n_nr: 5,  n_nr_miesz: 11);
71      adres2.Wypisz();
72      TAdres tablica_adresow[] = new TAdres[2];
73      tablica_adresow[0] = adres1;
74      tablica_adresow[1] = adres2;
75      for (int i = 1; i < tablica_adresow.length; i++) {
76          if (tablica_adresow[i] instanceof TAdres)
77              System.out.println("Wyswietlam element klasy TAdres");
78          if (tablica_adresow[i] instanceof TAdresSzczegolowy)
79              System.out.println("Wyswietlam element klasy TAdresSzczegolowy");
80
81          tablica_adresow[i].Wypisz();
82      }
83  }
```

2. Klasy i metody abstrakcyjne

```java
abstract class TZwierze
{
    String nazwa = "nie nazywa sie";
    TZwierze() {}
    TZwierze(String n_nazwa) { nazwa = n_nazwa; }
    abstract String dajTyp();
    abstract String dajGlos();
    String dajNazwe()
    {
        return nazwa;
    }
    void dajOpis() { System.out.println(dajTyp()+" "+dajNazwe()+" wydaje dźwięk "+dajGlos()); }
}
class TPies extends TZwierze
{
    TPies(){}
    TPies(String n_nazwa){ super(n_nazwa);}
    String dajTyp(){ return "Pies"; }
    String dajGlos(){ return "Wrrrr, Hau, Hau ";}
}
class TKot extends TZwierze
{
    TKot(){}
    TKot(String n_nazwa){ super(n_nazwa);}
    String dajTyp(){ return "Kot"; }
    String dajGlos(){ return "Miau, miau, mrrrr";}

}
public class Main {

    public static void main(String[] args) {
        //Zwierze mrowka = new Zwierze(); //Abstrakcyjność klasy zwierze oznacza, iż nie można  tworzy jej egzemplarzy (obiektów).
        TPies Burek = new TPies( n_nazwa: "Burek");
        Burek.dajOpis();
        TKot Filemon = new TKot( n_nazwa: "Filemon");
        Filemon.dajOpis();
    }
}
```

- 3 – klasa abstrakcyjna. Klasa w której zadeklarowano jakąkolwiek metodę abstrakcyjną jest klasą abstrakcyjną i musi być opatrzona specyfikatora **abstrac**t. Abstrakcyjność klasy zwierze oznacza, iż nie można tworzy jej egzemplarzy (obiektów).
- 11 - metoda abstrakcyjna nie ma implementacji (ciała) i powinna być zadeklarowana z modyfikatorem **abstract**.
- 13 - klasa taka może mieć zarówno metody abstrakcyjne jak i nie
- 22 - klasa dziedzicząca z klasy abstrakcyjnej musi zdefiniować wszystkie abstrakcyjne metody tej klasy, albo sama będzie klasą abstrakcyjną i wtedy jej definicja musi być opatrzona specyfikatorem **abstract**.
- 40 - abstrakcyjność klasy zwierze oznacza, iż nie można tworzyć jej egzemplarzy (obiektów).

3. Interfejsy

```
3  interface Pojazd
4  {
5      public void jedziemy(int predkosc);
6      public void stoimy();
7  }
8  class Samochod implements Pojazd {
9      @Override
10     public void jedziemy(int predkosc) { }
11     @Override
12     public void stoimy() { }
13     public void przyspieszamy() { }
14 }
15 class Rower implements Pojazd {
16     @Override
17     public void jedziemy(int predkosc) { }
18     @Override
19     public void stoimy() { }
20     public void niesiemy() { }
21 }
22 public class Main {
23     public static void main(String[] args) {
24         Samochod  Samochodzik = new Samochod();
25         Rower Rowerek = new Rower();
26     }
27 }
```

- 3 - przez interfejs rozumieć można taką specyficzną klasę abstrakcyjną, która z założenia zawiera tylko metody abstrakcyjne. Ponieważ wszystkie są abstrakcyjne, to w klasach, które go implementują wszystkie muszą być nadpisane. Interfejsy pozwalają "wymuszać" na klasah potomnych, aby posiadały niezbędne do współpracy z innymi klasami określone cechy. Klasy nie dziedziczą interfejsów, tylko je implementują. Interfejsy mogą dziedziczyć inne interfejsy. Interfejs jest jakby mocniej abstrakcyjną klasą, może mieścić jedynie metody, które z założenia są abstrakcyjne.
- 8 – klasa implementująca interfejs Pojazd

4. Interfejs dialogowy

```
3   class Osoba1
4   {
5       String nazwisko;
6       float srednia;
7       String uwagi;
8       static int ile_obiektow = 0;
9       public void Inicjuj() { ile_obiektow++; }
13      public void SetNazwisko(String lan)
14      {
15          nazwisko=lan;
16      }
17      public String GetNazwisko() { return nazwisko; }
21      public void SetUwagi(String lan) { uwagi=lan; }
25      public String GetUwagi() { return uwagi; }
29      public void SetSrednia(float srednia_) { srednia=srednia_; }
33      public float GetSrednia() { return srednia; }
37      public void Wstaw()
38      {
39          String S;
40          Inicjuj();
41          S = JOptionPane.showInputDialog( parentComponent: null,  message: "Podaj nazwisko");
42          SetNazwisko(S);
43          S = JOptionPane.showInputDialog( parentComponent: null,  message: "Podaj srednia");
44          SetSrednia(Float.parseFloat(S));
45          S =JOptionPane.showInputDialog( parentComponent: null,  message: "Podaj uwagi");
46          SetUwagi(S);
47      }
48      public void Wyswietl()
49      {
50          String napis="";
51          napis+="\n Nazwisko: "+nazwisko;
52          napis+="\n Średnia: "+srednia;
53          napis+="\n Uwagi: "+uwagi;
54          napis+="\n Liczba osób jest równa "+ile_obiektow;
55          JOptionPane.showMessageDialog( parentComponent: null, napis);
56      }
57      public boolean Szukaj(String s) { return nazwisko.equals(s); }
61  }
```

- 41 – wyświetlenie okienka dialogowego umożliwiającego podanie tekstu przez użytkownika. Ostatnim argumentem jest napis, który ma być w okienku wyświetlany.
- 55 – okienko wyświetlające statyczny tekst.

```java
public class Main {
    static int N=2;
    public static void main(String[] args) {
        Osoba1 Dane[] = new Osoba1[N];
        int ile=0;
        String s; char ch;
        do
        {
            s=JOptionPane.showInputDialog( parentComponent: null,  message: "Podaj wybor"
                    +"\n1 - Podaj dane kolejnej osoby,"
                    +"\n2 - Wyswietl dane osob"
                    +"\n3 - Wyszukaj osobe i wyswietl jej dane"
                    +"\nk - Koniec programu");
            ch = s.charAt(0);
            switch(ch)
            {
                case '1' : for (ile=0; ile<Dane.length;ile++)
                {
                    Dane[ile]=new Osoba1();
                    Dane[ile].Wstaw();
                }break;
                case '2' : for (int i=0; i<ile;i++)
                    Dane[i].Wyswietl();
                    break;
                case '3' : s = JOptionPane.showInputDialog( parentComponent: null, message: "Podaj nazwisko");
                    for (int i=0; i<ile;i++)
                        if (Dane[i].Szukaj(s))
                            Dane[i].Wyswietl();
                    break;
                case 'k' : JOptionPane.showMessageDialog( parentComponent: null,  message: "Koniec programu");
                    break;
                default : JOptionPane.showMessageDialog( parentComponent: null, message: "Zla opcja");
            }
        }while (ch != 'k') ;
        System.exit( status: 0);
    }

}
```

Zadanie 1 (35 pkt)

Napisz program, który przy wykorzystaniu interfejsu dialogowego umożliwi przeprowadzenie rozgrywki w „odchudzoną” wersję gry monopol. Uproszczenie gry polegać ma na likwidacji pól szansa/ryzyko oraz więzienie/idziesz do więzienia. Program ma być tak zaprojektowany, aby w możliwie maksymalnym stopniu wykorzystać możliwości Javy w zakresie dziedziczenia i szeroko pojętej obiektowości.