Nástroje balíčku java.util

ArrayList

- Dynamické pole

```
ArrayList<String> al = new ArrayList<String>();
al.add("Ahoj");
al.add("svete");
al.add("jak");
al.add("se");
al.add("mas");

System.out.println("al[2] = "+al.get(2) );

for(String s:al){
    System.out.println(s);
}

for(Iterator i = al.iterator(); i.hasNext();){
    String st = (String) i.next();
    System.out.println(st);
}
```

HashMap


```
HashMap<Integer,String> slovnik = new
                                  HashMap<Integer,String>();
slovnik.put(1, "jedna");
slovnik.put(2, "dva");
slovnik.put(3, "tri");

System.out.println("3 je "+slovnik.get(3));

for(Integer i: slovnik.keySet()){
    System.out.println("klic "+i+" je "+slovnik.get(i));
}

for(Entry e: slovnik.entrySet()){
    System.out.println("klic "+e.getKey()+" hodnota
"+e.getValue());
}
```

Zásobník - Zasobnik.java


```
public class Zasobnik {
    int [] data = new int[10];
    int ukazatel;

    public void push(int cislo){
        data[ukazatel++]=cislo;
    }

    public int pop(){
        return data[--ukazatel];
    }
}
```

Zásobník - Main


```
Zasobnik z = new Zasobnik();
// Stack<Integer> z = new Stack<Integer>();
z.push(5);
z.push(15);
z.push(30);
System.out.println(z.pop());
System.out.println(z.pop());
z.push(33);
System.out.println(z.pop());
System.out.println(z.pop());
```

Fronta - Fronta.java


```
public class Fronta {
    int [] data = new int[10];
    int pocatek = 0;
    int konec = 0;

    public int pop(){
        int n =  data[konec];
        konec = (konec + 1)%data.length;
        return n;
    }

    public void push(int cislo){
        data[pocatek] = cislo;
        pocatek = (pocatek + 1) % data.length;
    }
}
```

Fronta - Main


```
Fronta f = new Fronta();
f.push(1);
f.push(2);
f.push(3);
for (int i= 0; i < f.data.length; i++) {
    System.out.println("data["+i+"] = "+f.data[i]);
}
System.out.println(f.pop());
System.out.println(f.pop());
f.push(55);

for (int i= 0; i < f.data.length; i++) {
    System.out.println("data["+i+"] = "+f.data[i]);
}
System.out.println(f.pop());
System.out.println(f.pop());
```

Spojový seznam - Prvek.java


```
public class Prvek {
    int cislo;
    Prvek dalsi;

    public Prvek(int hodnota){
        this.cislo = hodnota;
    }

    public Prvek(int hodnota, Prvek predchozi){
        this.cislo = hodnota;
        this.dalsi = predchozi;
    }
}
```

Spojový seznam - Main


```
Prvek prvni = new Prvek(1,new Prvek(2,new Prvek(3,
                                        new Prvek(4))));
Prvek i = prvni;
while(i != null){
    System.out.println(i.cislo);
    i = i.dalsi;
}
Prvek jedenact = new Prvek(11,prvni.dalsi);
prvni.dalsi = jedenact;

i = prvni;
while(i != null){
    System.out.println(i.cislo);
    i = i.dalsi;
}
```

LinkedList

- Jiná implementace seznamu, přes spojový seznam

```
LinkedList<String> ll = new LinkedList<String>();
ll.add("prvni");
ll.add("druhy");
ll.addFirst("novyPrvni"); // zapise se pred ostatni
ll.add("treti");
ll.addLast("ctvrty");
for(String s:ll){
    System.out.println(s);
}
```

HashSet

- Matematická množina, bez duplikátů

```
String[] slova = {"Tento", "text", "obsahuje", "duplikovany",
"text"};
HashSet<String> hs = new HashSet<String>();
for(String s:slova){
    if(!hs.add(s)){
        System.out.println(s + " uz v mnozine slov je.");
    }
}
System.out.println("slovo " + (hs.contains("slovo")?"je":"neni") +
" v mnozine slov.");
System.out.println("obsahuje " +
(hs.contains("obsahuje")?"je":"neni") + " v mnozine slov.");
```

Dědičnost

- umožňuje postupné rozšiřování schopností objektů
- v Javě jednoduchá dědičnost
- klíčové slovo **extends**

Příklad:

- Geometrický objekt
  - Kruh
  - Obdélník
  - Pravidelný n-úhelník

GeometrickyObjekt.java


```
public class GeometrickyObjekt {
    protected double obsah;
    protected double obvod;

    public double getObsah() {
        return obsah;
    }

    public double getObvod() {
        return obvod;
    }

}
```

Kruh.java


```
public class Kruh extends GeometrickyObjekt {
    private double polomer;

    public Kruh(double polomer) {
        this.polomer = polomer;
        this.obsah = Math.PI*polomer*polomer;
        this.obvod = Math.PI*2*polomer;
    }

    @Override
    public String toString() {
        return "Kruh{" + "polomer=" + polomer + '}';
    }
}
```

Obdelnik.java


```
public class Obdelnik extends GeometrickyObjekt{
    private double sirka;
    private double vyska;

    public Obdelnik(double sirka, double vyska) {
        this.sirka = sirka;
        this.vyska = vyska;

        this.obsah = sirka*vyska;
        this.obvod = 2 * (sirka+vyska);
    }

    public double getSirka() {
        return sirka;
    }

    public double getVyska() {
        return vyska;
    }
}
```

Použití


```
Kruh k = new Kruh(1);
System.out.println(k);
System.out.println("Obsah kruhu je" + k.getObsah());

Obdelnik o = new Obdelnik(33.3, 22.2);
System.out.println("Obsah obdelnika je "+o.getObsah());
```

- V třídách Kruh ani Obdelnik není metoda getObsah naimplementována. Ale můžeme ji volat, neboť se do objektů zdědila.
- Stejně tak můžeme ve třídách Obdelnik a Kruh přistupovat k proměnným obsah a obvod.
- Klíčové slovo **protected** je něco mezi **private** a **public**. Umožňuje třídám z jednoho balíčku přístup.

Abstraktní metody a třídy

- Pokud víme, že nějakou metodu mají nabízet všechny třídy, které od nás dědí, ale nevíme, jak ji implementovat, protože je to závislé na dané třídě, můžeme naši metodu udělat abstraktní.

```
public abstract class GeometrickyObjekt {
    protected double obsah;
    protected double obvod;

    public double getObsah() {
        return obsah;
    }

    public double getObvod() {
        return obvod;
    }

    public abstract void vykresliNaRadku(char znak);
}
```

Úprava kódu třídy Obdelnik


```
public class Obdelnik extends GeometrickyObjekt{
    private double sirka;
    private double vyska;

    public Obdelnik(double sirka, double vyska) {
        this.sirka = sirka;
        this.vyska = vyska;
        this.obsah = sirka*vyska;
        this.obvod = 2 * (sirka+vyska);
    }

    public double getSirka() {
        return sirka;
    }

    public double getVyska() {
        return vyska;
    }

    @Override
    public void vykresliNaRadku(char znak) {
        for (int i= 0; i < this.vyska; i++) {
            for (int j= 0; j < this.sirka; j++) {
                System.out.print(znak);
            }
            System.out.println("");
        }
    }
}
```

Interface

- Třídu obsahující jen abstraktní metody lze nazvat interface

```
public interface Auto {
    String getSpz();
    double getObjem();
    int getPocetKoni();
}
```

je stejné jako

```
public abstract class Auto{
    public abstract String getSpz();
    public abstract double getObjem();
    public abstract int getPocetKoni();
}
```

- Interface se implementuje, nedědí, používá se tedy klíčové slovo **implements**

Enum

- Výčtový typ java zavádí až od verze 5

```
public enum TypPaliva {
    BENZIN,
    NAFTA,
    LPG,
    CNG
}
```

- Práce s ním jako s konstantami typu int, lze je použít ve switchi, více možností jako v jazyce C/C++