Objekty

Objekty I

- objekt reprezentuje něco skutečného
- objekt má vlastnosti
  - proměnné
    - veřejné - vidí je všichni, mohou je i měnit
    - **privátní** - zná je jen objekt
- objekt má schopnosti
  - metody
    - **veřejné** - mohou je využít všichni, kteří jej znají
    - privátní - může je využít jen objekt sám, ostatní je neznají

Objekty II

Třída

- shrnuje vlastnosti
- obecný popis objektu
- v Javě začíná velkými písmeny
  - public class Auto { ... }
  - public class Osoba { ... }

Instance

- konkrétní objekt
  - Auto a = new Auto(spz, barva, motor);
  - Osoba o = new Osoba(jmeno, prijmeni, rc);

Obdélník

Vytvořte třídu Obdelnik, která bude mít 2 skryté instanční proměnné - šířku a výšku. Třída bude mít 2 veřejné metody, jedna zjistí obsah, druhá obvod.

Obdélník - řešení


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

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

    public double getObvod(){
        return (this.sirka+this.vyska)*2;
    }

    public double getObsah(){
        return this.sirka*this.vyska;
    }
}
```

Obdélník pokračování

Každý objekt obsahuje metodu `toString();` Ta slouží k tomu, aby bylo možné zapsat informaci o objektu do výstupu v lidsky čitelné podobě.

Napište metodu toString do třídy Obdelnik tak, aby dávala smysluplnější výpis než nyní.

Řešení


```
@Override
public String toString() {
    return "Obdelnik{sirka="+sirka+" vyska="+vyska+"}";
}
```

Anotace `@Override` pochází z Javy verze 5 a není povinná, dává nám informaci o tom, že přepisujeme metodu, která již dříve nějak fungovala.

Pokračování

Pokud bychom chtěli srovnávat, zda jsou dva obdélníky stejné, pak nemůžeme použít operátor ==

```
obdelnikA == obdelnikB
```

V tomto případě bychom většinou dostali negativní odpověď. Důvodem je, že se porovnávají reference, tedy adresy objektů v paměti.

Z tohoto důvodu je nutné napsat vlastní metodu

```
public boolean equals(Obdelnik o)
```

která provede porovnání objektů podle rozumnějších vlastností, zde šířky a výšky.

Řešení


```
public boolean equals(Obdelnik o){
   return o.sirka == this.sirka && o.vyska == this.vyska;
}
```

Pokud bychom psali program tak, aby byl kompatibilní s knihovními funkcemi, museli bychom napsat metodu např. takto:

```
@Override
public boolean equals(Object obj) {
    boolean equals = false;
    if(obj instanceof Obdelnik){
        Obdelnik o = (Obdelnik) obj;
        equals = (o.sirka == this.sirka && o.vyska ==
this.vyska);
    }
    return equals;
}
```

Další modifikace

Do třídy Obdélník přidejte konstruktor, který bude tvořit čtverce, tedy obdélníky se stejnou výškou i šířkou.

Do třídy Obdélník přidejte jednu statickou proměnnou počet a jednu instanční proměnnou název, upravte konstruktor tak, aby obdélníky pojmenovával stylem Obdelnik<číslo>, kde číslo je počet obdélníků dosud vytvořených.

Řešení


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

    private String jmeno;
    private static int pocet = 0;

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

        this.jmeno = "Obdelnik"+pocet+1;
        pocet++;
    }
    public Obdelnik(double sirka){
        this(sirka, sirka);
    }
    public String getJmeno() {
        return jmeno;
    }
...
}
```

Komplexní čísla

Napište program, který vytvoří uživatelem zadaný počet náhodně vygenerovaných komplexních čísel, uloží je do pole a následně je vypíše.

Komplexní čísla - řešení


```
public class Complex {
    private double real;
    private double imag;

    public Complex(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }
    @Override
    public String toString() {
        return real + (imag < 0 ? " - " : " + ")+
Math.abs(imag) +"i";
    }
}
```

Komplexní čísla - řešení II


```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Pocet cisel: ");
    int a = s.nextInt();
    Complex[] pole = new Complex[a];
    for (int i = 0; i < pole.length; i++) {
        pole[i] = new Complex(Math.random(),
Math.random());
    }
    for (int i = 0; i < pole.length; i++) {
        Complex complex = pole[i];
        System.out.println(complex);
    }
}
```