Rešitev šeste domače naloge (Točka in premica)

Razred `Tocka`

Pri atributih ni dilem. Vsak objekt tipa `Tocka` bo imel dva: koordinati x in y.

```
import java.util.Locale;

public class Tocka {

    private double x, y;
    ...
}
```

Konstruktor enostavno skopira parametra v pripadajoča atributa:

```
public Tocka(double x, double y) {
    this.x = x;
    this.y = y;
}
```

»Getterja« sta trivialna:

```
public double vrniX() {
    return this.x;
}

public double vrniY() {
    return this.y;
}
```

Metoda `toString` si pomaga z metodo `String.format`:

```
public String toString() {
    return String.format(Locale.US, "(%.2f, %.2f)", this.x, this.y);
}
```

S parametrom `Locale.US` zagotovimo, da se bo kot decimalno ločilo ne glede na sistemske jezikovne nastavitve uporabila pika.

Metoda `izhodisce` vrne objekt, ki predstavlja točko (0, 0). Da ne bomo ob vsakem klicu metode ustvarjali nove kopije tega objekta, ga bomo deklarirali in ustvarili kot statični nespremenljivi atribut:

```
public class Tocka {
    private static final Tocka IZHODISCE = new Tocka(0, 0);
    ...
    public static Tocka izhodisce() {
        return Tocka.IZHODISCE;
    }
    ...
}
```

Razdaljo med točkama (x_1, y_1) in (x_2, y_2) izračunamo po formuli $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$:

```
public double razdalja(Tocka t) {
    double dx = t.x - this.x;
    double dy = t.y - this.y;
    return Math.sqrt(dx * dx + dy * dy);
}
```

Pri metodi `razdaljaOdIzhodisca` si lahko pomagamo z metodama `razdalja` in `izhodisce`:

```
public double razdaljaOdIzhodisca() {
    return this.razdalja(Tocka.izhodisce());
}
```

Razred `Premica`

Začetni del razreda lahko napišemo skoraj miže ...

```
import java.util.Locale;

public class Premica {

    private double k, n;

    public Premica(double k, double n) {
        this.k = k;
        this.n = n;
    }

    public double vrniK() {
        return this.k;
    }

    public double vrniN() {
        return this.n;
    }

    public String toString() {
        return String.format(Locale.US, "y = %.2f x + %.2f", this.k, this.n);
    }
    ...
}
```

Če vemo, da točka (x_0, y_0) leži na premici $y = kx + n$, in če poznamo koordinato x_0, potem koordinato y_0 izračunamo tako, da x_0 vstavimo v enačbo premice ($y_0 = kx_0 + n$):

```
public Tocka tockaPriX(double x) {
    return new Tocka(x, this.k * x + this.n);
}
```

Če poznamo smerni koeficient premice $y = kx + n$ in če vemo, da ta potuje skozi točko (x_0, y_0), potem lahko manjkajoči parameter n izračunamo kot $n = y_0 - kx_0$:

```
public static Premica skoziTocko(double k, Tocka t) {
    return new Premica(k, t.vrniY() - k * t.vrniX());
}
```

Vzporednica premice $y = kx + n$ ima prav tako smerni koeficient k. Ker vemo, skozi katero točko potuje, si lahko pomagamo z metodo `skoziTocko`:

```
public Premica vzporednica(Tocka t) {
    return Premica.skoziTocko(this.k, t);
}
```

Smerni koeficient pravokotnice na premico $y = kx + n$ pa je enak $-1 \,/\, k$:

```
public Premica pravokotnica(Tocka t) {
    return Premica.skoziTocko(-1.0 / this.k, t);
}
```

Če premici $y = k_1x + n_1$ in $y = k_2x + n_2$ nista vzporedni, lahko koordinato x njunega presečišča izračunamo tako, da izenačimo obe desni strani njunih enačb in iz nastale enačbe izračunamo x:

$$\begin{aligned} k_1x + n_1 &= k_2x + n_2 \\ (k_1 - k_2)x &= n_2 - n_1 \\ x &= (n_2 - n_1) \,/\, (k_1 - k_2) \end{aligned}$$

Koordinato y presečišča pa lahko pridobimo s pomočjo metode `tockaPriX`. Metodo lahko pokličemo nad premico `this` ali `p` (obakrat dobimo isti rezultat):

```
public Tocka presecisce(Premica p, double epsilon) {
    if (Math.abs(this.k - p.k) < epsilon) {
        return null;
    }
    double x = (p.n - this.n) / (this.k - p.k);
    return this.tockaPriX(x);
}
```

Pravokotna projekcija točke T na premico p je presečišče premice p in pravokotnice na premico p skozi točko T:

```
public Tocka projekcija(Tocka t) {
    Premica pravokotnica = this.pravokotnica(t);
    return this.presecisce(pravokotnica, 0.0);
}
```

Ker premica ne more biti vzporedna pravokotnici nase, nastavimo parameter `epsilon` na vrednost 0.

Razdalja med premico p in točko T je enaka razdalji med točko T in pravokotno projekcijo točke T na premico p:

```
public double razdalja(Tocka t) {
    return t.razdalja(this.projekcija(t));
}
```

Razdalja med premico in izhodiščem je zgolj poseben primer razdalje med premico in točko:

```
public double razdaljaOdIzhodisca() {
    return this.razdalja(Tocka.izhodisce());
}
```

Razdaljo med vzporednima premicama p in q izračunamo tako, da si na premici q izberemo točko (npr. točko s koordinato $x = 0$), nato pa izračunamo razdaljo med premico p in izbrano točko:

```
public double razdalja(double n) {
    Premica vzporednica = new Premica(this.k, n);
    Tocka t = vzporednica.tockaPriX(0.0);
    return this.razdalja(t);
}
```