Skapa nya objekttyper

En modell av en polylinje – polylinjens hörn ska lagras i en vektor

A) En modell av en punkt i planet

En punkt i planet har sitt namn och sina koordinater. Punktens koordinater är hela tal.

En punkt kan bestämmas med ett namn och två heltalskoordinater. Det går även att bestämma en punkt utifrån en annan punkt – en kopia kan skapas.

Man kan skapa en teckensträng som representerar en punkt. Denna teckensträng kan vara på formen `(A 3 4)`. En punkts namn och koordinater kan erhållas. Koordinaterna kan ändras, en koordinat i taget. Avståndet mellan två givna punkter kan bestämmas. Man kan kontrollera huruvida två givna punkter är likadana eller inte.

En modell av en punkt i planet ska skapas: man ska skapa en definitionsklass som heter `Punkt`.

Ett enkelt testprogram för klassen Punkt

```
import java.io.*;    // PrintWriter

class PunktTest
{
    public static void main (String[] args)
    {
        PrintWriter    out = new PrintWriter (System.out, true);

        // testa en konstruktor och en transformator
        Punkt    p1 = new Punkt ("A", 3, 4);
        Punkt    p2 = new Punkt ("B", 5, 6);
        out.println (p1 + "    " + p2);

        // testa inspektorer
        String    n = p1.getNamn ();
        int    x = p1.getX ();
        int    y = p1.getY ();
        out.println (n + " " + x + " " + y);

        // testa en kombinator och en komparator
        double    d = p1.avstand (p2);
        out.println (d);
        boolean    b = p1.equals (p2);
        out.println (b);

        // testa mutatorer
        p2.setX (1);
        p2.setY (2);
        out.println (p2);

        // testa en konstruktor till
        Punkt    p = new Punkt (p1);
        out.println (p);
    }
}
```

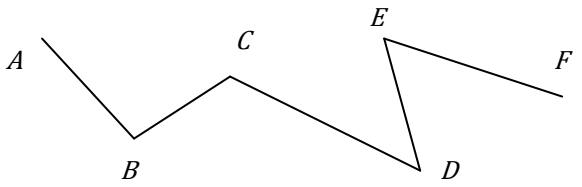
Uppgifter i samband med punkter i planet

1. Skapa en definitionsklass som heter `Punkt`, som representerar en punkt i planet. Kör testprogrammet `PunktTest` i samband med den nyskapade klassen.

2. Skapa en bild som visar hur avståndet mellan två punkter i planet bestäms. Relatera denna bild till koden i den motsvarande metoden: för in variablernas namn i bilden.

B) En polylinje i planet

En polylinje är en geometrisk figur som består av en serie bundna linjesegment. Ändpunkter för dessa segment utgör polylinjens hörn. En polylinje kan föreställas så här:



En polylinje kan bestämmas med sina hörn, sin färg och sin bredd. En tom polylinje saknar hörn.

Man kan skapa en teckensträng som representerar en polylinje. Denna teckensträng kan vara på formen `{[(A 3 4)(B 1 2)(C 2 3)(D 5 1)], svart, 1}`. En polylinjes hörn, färg, bredd och längd kan erhållas. Färgen och bredden kan ändras. En polylinjes utseende ändras även när dess hörnsekvens ändras. Det går att lägga till ett nytt hörn till polylinjen – antingen på slutet, eller framför ett hörn vars namn är givet. Det går även att ta bort ett hörn med ett givet namn.

En modell av en polylinje i planet ska skapas: man ska skapa en definitionsklass som heter `Polylinje`.

En modell av en polylinje – ej fullständig

```
public class Polylinje
{
    private Punkt[]    horn;
    private String     farg = "svart";
    private int        bredd = 1;

    public Polylinje ()
    {
        this.horn = new Punkt[0];
    }

    public Polylinje (Punkt[] horn)
    {
        this.horn = new Punkt[horn.length];
        for (int i = 0; i < horn.length; i++)
            this.horn[i] = new Punkt (horn[i]);
    }

    public String toString () {}

    public Punkt[] getHorn () {}

    public String getFarg () {}

    public int getBredd () {}

    public void setFarg (String farg) {}

    public void setBredd (int bredd) {}

    public double langd () {}

    public void laggTill (Punkt horn)
    {
        Punkt[]    h = new Punkt[this.horn.length + 1];
```

```
        int    i = 0;
        for (i = 0; i < this.horn.length; i++)
            h[i] = this.horn[i];
        h[i] = new Punkt (horn);

        this.horn = h;
    }

    public void laggTillFramfor (Punkt horn, String hornNamn) {}

    public void taBort (String hornNamn) {}
}
```

Uppgifter i samband med polylinjer

1. Komplettera definitionsklassen `Polylinje`: implementera metoderna och kommentera klassen samt dess medlemmar. I metoden `laggTillFramfor` ska det hörn som parameterreferensen refererar till kopieras. I metoden `getHorn` ska en vektor som innehåller kopior av polylinjens hörn skapas, och en referens till denna vektor ska returneras.

2. Skapa en definitionsklass till: `Polylinje1`. I denna klass ska parameterreferenser kopieras, i stället för de refererade resurserna. I metoden `getHorn` ska referensen till egna hörn returneras.

Vilken strategi är bättre: att kopiera resurser (som i klassen `Polylinje`) eller bara referenser (som i klassen `Polylinje1`)?

3. Skapa ett enkelt testprogram, `PolylinjeTest`, som använder konstruktorerna och metoderna i klassen `Polylinje`.

4. Skapa en bild som representerar ett objekt av typen `Polylinje`. Bland annat ska bilden innehålla objektet och dess referenser, den refererade vektorn och dess referenser, samt de hörn som refereras ifrån vektorn. Bilden ska förses med rätta beteckningar.

5. Åskådliggör den algoritm som används i metoden `laggTillFramfor`. Man ska rita en serie bilder som visar hur det givna hörnet så småningom förs in på rätt ställe i den aktuella polylinjen. Bilderna ska förses med rätta beteckningar.

6 Ett objekt av typen `Polylinje` har sin egen vektor, där polylinjens hörn lagras. Är denna strategi minneseffektiv? Är den tidseffektiv? Vad händer om man ofta lägger till eller tar bort hörn?

C) Skapa och använda polylinjer

I ett program, som heter `ValjPolylinje`, skapar man och använder polylinjer av typen `Polylinje`.

Ett antal slumpmässiga polylinjer skapas och visas. En polylinje har ett slumpmässigt antal hörn, och färgen är antingen blå, röd eller gul. Namn på polylinjens hörn är stora bokstäver i engelska alfabetet. Namnen bestäms slumpmässigt, men två hörn kan inte ha likadana namn. Programmet bestämmer och visar den kortaste av de gula polylinjerna.

Programmet `ValjPolylinje` har följande struktur:

```
class ValjPolylinje
{
    public static final Random    rand = new Random ();
    public static final int    ANTAL_POLYLINJER = 10;

    public static void main (String[] args)
    {
        // skapa ett antal slumpmässiga polylinjer
        Polylinje[]    polylinjer = new Polylinje[ANTAL_POLYLINJER];
        for (int i = 0; i < ANTAL_POLYLINJER; i++)
            polylinjer[i] = slumpPolylinje ();

        // visa polylinjerna

        // bestäm den kortaste av de polylinjer som är gula

        // visa den valda polylinjen
    }
```

```
    // slumpPunkt returnerar en punkt med ett slumpmässigt namn, som är en stor bokstav i
    // det engelska alfabetet, och slumpmässiga koordinater.
    public static Punkt slumpPunkt ()
    {
        String    n = "" + (char) (65 + rand.nextInt (26));
        int    x = rand.nextInt (11);
        int    y = rand.nextInt (11);

        return new Punkt (n, x, y);
    }

    // slumpPolylinje returnerar en slumpmässig polylinje, vars färg är antingen blå, eller röd
    // eller gul. Namn på polylinjens hörn är stora bokstäver i det engelska alfabetet. Två hörn
    // kan inte ha samma namn.
    public static Polylinje slumpPolylinje ()
    {
        // skapa en tom polylinje, och lägg till hörn till den
        Polylinje    polylinje = new Polylinje ();
        int    antalHorn = 2 + rand.nextInt (7);
        int    antalValdaHorn = 0;
        boolean[]    valdaNamn = new boolean[26];
        // ett och samma namn kan inte förekomma flera gånger
        Punkt    valdPunkt = null;
        char    valtChar = 0;
        while (antalValdaHorn < antalHorn)
        {

        }


        // sätt färg
    }
}
```

Uppgift i samband med användning av polylinjer

Komplettera och prova programmet `ValjPolylinje`. Vad händer om ingen av polylinjerna är gul?

D) En iterator till en polylinje

En iterator till en polylinje kan definieras och implementeras. Med en sådan iterator kan hörnen i polylinjen besökas och användas i tur och ordning.

En iterator till en polylinje kan definieras och implementeras i en inre klass i klassen `Polylinje`. Denna klass kan heta `PolylinjeIterator`, och kan se ut så här:

```
public class PolylinjeIterator
{
    private int    aktuell = -1;

    public PolylinjeIterator ()
    {
        if (Polylinje.this.horn.length > 0)
            aktuell = 0;
    }

    public boolean finnsHorn ()
    {
        return aktuell != -1;
    }

    public Punkt horn () throws java.util.NoSuchElementException
    {
        if (!this.finnsHorn ())
            throw new java.util.NoSuchElementException (
                                "slut av iterationen");
```

```
        Punkt    horn = Polylinje.this.horn[aktuell];

        return horn;
    }

    public void gaFram ()
    {
        if (aktuell >= 0  &&
            aktuell < Polylinje.this.horn.length - 1)
            aktuell++;
        else
            aktuell = -1;
    }
}
```

Uppgifter i samband med iteratorer

1. Skapa en iterator (ett objekt av typen `PolylinjeIterator`) i testprogrammet för klassen `Polylinje`. Besök med denna iterator polylinjens hörn i tur och ordning, och skriv ut dem.

2- Visualisera en iteration genom en polylinje. Följ ändringarna i iterator-objektet.