```java
package w12pruefung.v1;
import java.util.*;
import w12pruefung.Punkt;

public class Polygon {
    protected final List<Punkt> mPunkte;
    public Polygon(Punkt erster, Punkt... weitere) {
        this();
        mPunkte.add(erster);
        for (Punkt p : weitere)
            if (p != erster) mPunkte.add(p);
    }
    private Polygon() {
        mPunkte = new ArrayList<Punkt>();
    }
    public int anzahlPunkte() {
        return mPunkte.size();
    }
    public Punkt anfangsPunkt() {
        return mPunkte.get(0);
    }
    private Punkt punkt(int nr) {
        int n = nr % anzahlPunkte();
        return mPunkte.get(n);
    }
    public boolean equals(Polygon p) {
        if (anzahlPunkte() != p.anzahlPunkte()) return false;
        int nr = mPunkte.indexOf(p.anfangsPunkt());
        if (nr == -1) return false;
        for (int i = 0; i < anzahlPunkte(); i++)
            if (!punkt(i + nr).equals(p.punkt(i))) return false;
        return true;
    }
    public int hashCode() {
        return mPunkte.hashCode();
    }
}
```

```java
package w12pruefung.v2;
import java.util.*;
import w12pruefung.Punkt;

public final class Polygon {
    private final List<Punkt> mPunkte;
    public Polygon(Punkt erster, Punkt zweiter, Punkt... weitere) {
        this();
        mPunkte.add(erster);
        mPunkte.add(zweiter);
        for (Punkt p : weitere) mPunkte.add(p);
        Punkt letzter = weitere[weitere.length - 1];
        if (!erster.equals(letzter)) mPunkte.add(erster);
    }
    private Polygon() {
        mPunkte = new ArrayList<Punkt>();
    }
    public int anzahlPunkte() {
        return mPunkte.size() - 1;
    }
    public Punkt anfangsPunkt() {
        return mPunkte.get(0);
    }
    public boolean equals(Polygon p) {
        return p.mPunkte.equals(mPunkte);
    }
    public boolean equals(Object o) {
        if (o == this) return true;
        if (o == null) return false;
        Polygon p = (Polygon) o;
        return p.equals(this);
    }
    public int hashCode() {
        return anzahlPunkte();
    }
}
```

```java
package w12pruefung.v3;
import w12pruefung.Punkt;

public class PolygonPunkt extends Punkt {
    private PolygonPunkt mVoriger;
    public PolygonPunkt(Punkt p, PolygonPunkt voriger) {
        super(p.x(), p.y());
        mVoriger = voriger;
    }
    public PolygonPunkt voriger() {
        if (mVoriger == null) throw new IllegalStateException();
        return mVoriger;
    }
    void schliessen() {
        PolygonPunkt a = this;
        while (a.mVoriger != null) a = a.mVoriger;
        a.mVoriger = this;
    }
    public Punkt anfangsPunkt() {
        return this;
    }
}
```

```java
package w12pruefung.v3;
import w12pruefung.Punkt;

public class Polygon extends PolygonPunkt {
    public Polygon(Punkt erster, Punkt... weitere) {
        super(erster, null);
        PolygonPunkt voriger = this;
        for (Punkt p : weitere) voriger = new PolygonPunkt(p, voriger);
        voriger.schliessen();
    }
    public int anzahlPunkte() {
        int erg = 1;
        PolygonPunkt v = this;
        for (PolygonPunkt p = v.voriger(); p != v; p = p.voriger()) erg++;
        return erg;
    }
}
```

```java
package w12pruefung.v4;
import w12pruefung.*;

public class PolygonPunkt extends Punkt {
    final int mNummer;
    public PolygonPunkt(Punkt p, int n) {
        super(p.x(), p.y());
        mNummer = n;
    }
}
```

```java
package w12pruefung.v4;
import w12pruefung.*;
import java.util.*;

public class Polygon {
    private final Set<PolygonPunkt> mPunkte = new HashSet<PolygonPunkt>();
    public Polygon(Punkt... punkte) {
        for (Punkt p : punkte) {
            int nr = mPunkte.size();
            PolygonPunkt pp = new PolygonPunkt(p, nr);
            mPunkte.add(pp);
        }
    }
    public boolean equals(Object o) {
        if (o == this) return true;
        if (o == null) return false;
        if (o.getClass() != getClass()) return false;
        Polygon p = (Polygon) o;
        return p.mPunkte.equals(mPunkte);
    }
    public int anzahlPunkte() {
        return mPunkte.size();
    }
    public Punkt anfangsPunkt() {
        for (PolygonPunkt p : mPunkte) return p;
        throw new InternalError();
    }
}
```