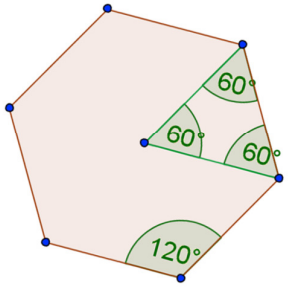
EF Informatik **Lösungsblatt PolyForms 4**

Thema: Objektorientiertes Entwerfen / Modellieren / Implementieren mit Java

Teil a): ***Entwurf in Pseudocode***

```
private zeichneSechseck()
  stift hoch
  bewege stift bis (xpos | ypos)
  drehe stift bis angle + 120°
  bewege stift um size
  drehe stift bis angle
  stift runter
  wiederhole 6-mal
    bewege stift um size
    drehe stift um -60°
```



Teil b): ***Implementierung in Java***

```
private void zeichneSechseck()
{
  stift.hoch();
  stift.bewegeBis(xpos, ypos);
  stift.dreheBis(angle + 120);
  stift.bewegeUm(size);
  stift.dreheBis(angle);
  stift.runter();

  for(int i = 0; i < 6; i++)
  {
     stift.bewegeUm(size);
     stift.dreheUm(-60);
  }
}
```

Zentraler Aspekt bei der parameterisierten öffentlichen Methode *zeichneSechseck(…)* ist die Wiederverwendung der privaten, parameterlosen Variante wie oben gezeigt in Verbindung mit dem vorhergehenden Setzen der Klassenattribute auf die Werte der Aufrufparameter sowie der Kennnummer für die Form *Sechseck*. Hierdurch werden einerseits Redundanzen und damit Fehlermöglichkeiten vermieden sowie andererseits Konsistenz zwischen der angezeigten Form und den Attributwerten der aktuellen Klasseninstanz der Klasse *PolyForms* sichergestellt.

```
public void zeichneSechseck(int px, int py, int ps)
{
  setXpos(px);
  setYpos(py);
  setSize(ps);
  setShape(2);

  zeichneSechseck();
}
```

Teil c): Methode *zeichne()* / *Java-Quellcode*

Da jetzt zwei Formen (Dreieck / Sechseck) unterschieden werden müssen, ist entsprechend in der Methode *zeichne()* abhängig von der aktuell gesetzten Kennnummer eine Fallunterscheidung bzw. Verzweigung erforderlich.

```
public void zeichne()
{
   if (shape == 1)
       zeichneDreieck();
   else
       zeichneSechseck();
}
```