NACHNAME:	**SEMESTER:** ☐ **M5** ☐ **M6** ☐ M3 ☐ M4 ☐ M7
VORNAME:	**VERTIEFUNG:** ☐ **FV** ☐ **IM**

ANLAGE, HILFSMITTEL:	• Ausdruck des vorab bekannt gemachten Quelltextes

UNBEDINGT BEACHTEN:	• **Bevor** Sie mit der Bearbeitung beginnen, müssen die Angaben zur Person auf dieser Seite **vollständig** ausgefüllt sein.

GENERELLE VORGABEN:	• Es sind keinerlei Kommentare verlangt, weder `javadoc`-Kommentare noch andere. • Programmier-Richtlinien (insbesondere Checkstyle) sind zu beachten. Bei Testklassen dürfen aber *magic numbers* verwendet werden.

Aufgabe 1: (19 Punkte)

Die Aufgabe nimmt Bezug auf alle drei Versionen von `Polynom`.

Geben Sie für die drei Versionen (kurz **V1**, **V2**, **V3**) jeweils an,
ob die Anweisung in Ordnung (**OK**) ist, zu einem Compile-Fehler (**CF**) führt oder zur Laufzeit eine Exception (**E**) wirft.

a) Prüfen Sie die angegebenen Anweisungen.

	V1	V2	V3
`Polynom p1 = new Polynom();`			
`Polynom p2 = new Polynom(7/8);`			
`Polynom p3 = new Polynom(0, 0, 7);`			
`Polynom p4 = new Polynom(1, 1, 1, 1, 1);`			
`Polynom p5 = new Polynom(1) { };`			
`Polynom p6 = new Polynom(0, 0, 7) { };`			

b) Prüfen Sie nun die Anweisung

```
Polynom p = new Polynom(E);
```

wobei `E` zuvor über den jeweils angegebenen statischen Import bekannt gemacht wird.

	V1	V2	V3
`import static java.lang.Math.*;`			
`import static pruefung.Funktionen.*;`			
`import static pruefung.Funktion.*;`			
`import static pruefung.StammFunktion.*;`			

Aufgabe 2: (14 Punkte)

Die Aufgabe bezieht sich auf das `Polynom` der `version1`.

a) Die Testmethode `testeEqualsEinfach` geht schief. Durch Änderung einer einzigen Methode läuft der Test durch. Geben Sie diese Methode vollständig an.

b) Auch der Test `testeUnveraenderbarkeitKonstruktor` führt zu einem Fehler. Beheben Sie diesen Fehler. Geben Sie den verbesserten Konstruktor vollständig an.

Aufgabe 3: (19 Punkte)

Die Aufgabe bezieht sich auf `version2` der Klasse `Polynom`.

Die Testmethode `testeSumme` führt zu einem Fehler. Die Ursache liegt an einem prinzipiellen Problem der Klasse, das immer dann entsteht, wenn der Koeffizient der höchsten Potenz 0 ist.

a) Schreiben Sie eine neue Testmethode (in `PolynomTest`), welche für zwei solche Polynome mit „führenden Nullen“ den Grad überprüft. Eines der Polynome soll eine führende Null und den Grad -1 haben, das andere zwei führende Nullen und den Grad 0.

b) Die beste Behebung des Fehlers ist, Polynome bereits in „Normalform“, also ohne führende Nullen zu erzeugen. Nehmen Sie diese Änderung vor. Geben Sie betroffene Methoden / Konstruktoren vollständig an.

Aufgabe 4: (19 Punkte)

Die Aufgabe nimmt Bezug auf `version1` und `version2` von `Polynom`.

In `version2` besitzt die Klasse `Polynom` eine in vielerlei Hinsicht wesentlich bessere Methode `stammFunktion` als in `version1`.

a) Der vorhandene Test für die Stammfunktion prüft nur einzelne Funktionswerte und das sogar nur näherungsweise. Schreiben Sie zunächst eine entsprechende Testmethode `testeStammfunktionExakt`, welche die komplette Funktion exakt prüft.

b) Überschreiben Sie nun die Methode `stammFunktion` in `version1` der Klasse `Polynom` entsprechend. Obiger Test muss dann fehlerfrei durchlaufen.

Aufgabe 5: (19 Punkte)

Die Aufgabe benutzt alle drei Versionen der Klasse `Polynom`.

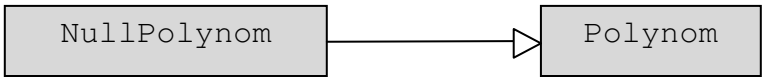
a) Die statische Variable `E` in `UeberallDefinierteFunktion` werde (statt auf 1000) auf 0 gesetzt. Die Variablen `p1`, `p2`, `p3` und `f1` seien wie folgt initialisiert:

```
Polynom p1 = new Polynom(86);
Polynom p2 = new Polynom(86);
Polynom p3 = new Polynom(1, 86);
Funktion f1 = p1;
```

Geben Sie an, welches Ergebnis (`true` oder `false`) die jeweiligen Aufrufe liefern.

	V1	V2	V3
`f1.equals(p1)`			
`p2.equals(f1)`			
`f1.equals(p3)`			
`p3.equals(f1)`			
`p3.equals(p1)`			

b) Betrachten Sie folgende Beziehung:



Ergänzen Sie (in den eckigen Klammern):

Bei der Beziehung handelt es sich um eine [ ]. Diese ist nur möglich bei der Version [ ] von `Polynom`, weil die Klasse `Polynom` in den anderen beiden Versionen [ ] ist. Die Beziehung bedeutet: Ein `NullPolynom` [ ] `Polynom`. Dies scheint korrekt zu sein. Ein [ ] kann dann aber überall verwendet werden, wo ein Objekt der anderen Klasse verwendet werden kann. Dies nennt man das [ ]. Wenn die Klasse `Polynom` nicht [ ] ist, kann das zu einem ernsten Problem führen.