1. Einleitung

TL;DR; In dieser Übung geht es um das Java Objekt „BigInteger“, das Interface „Comparable“ sowie um das Implementieren von eigenen Funktionen („POW“ und „probablePrime“).

In folgender Übung beschäftigen wir uns mit Objektorientierung, deren Anwendung und die Implementierung von Interfaces. Konkret geht es um die Klasse: java.math.BigInteger. Diese Klasse implementiert unter anderem das Interface: java.lang.Comparable. Zum Bearbeiten der Übung werden weitere Klassen benötigt, welche als Vorgabe noch importiert werden müssen. Abschließend wird eine eigene Versionen der Methoden java.math.pow und java.math.BigInteger.probablePrime implementiert.

2. BigInteger und das Interface Comparable

Die Java Klasse „BigInteger“ implementiert unter anderem das Interface „Comparable“. Informieren Sie sich in der Java Dokumentation über die Anwendung. Anschließend überlegen Sie sich, welches Ergebnis unter Berücksichtigung der funktionsweise bei folgenden Aufrufen ausgegeben wird:

Gegeben:

- BigInteger ***bigIntA*** = 5
- BigInteger ***bigIntB*** = 0

Gesucht:

Aufruf	Ergebnis
bigIntA.compareTo(bigIntB)	
bigIntB.compareTo(bigIntA)	
bigIntA.compareTo(bigIntA)	

Wichtig: Das Verhalten bzw. die Funktionsweise von BigInteger.compareTo() wird später benötigt, um die Primzahlen Prüfung zu Programmieren.

Informationen über java.math.BigInteger, das Interface java.lang.Comparable sowie dessen Verwendung können Sie auf diversen Webseiten nachschlagen. In den meisten Fällen ist Java ist auch eine Insel (http://openbook.rheinwerk-verlag.de/javainsel9/) eine gute Anlaufstelle.

Hinweis: Es steht Ihnen frei die Vergleiche in einer Anwendungsklasse abzubilden.

3. Math.pow in Eigenregie

Wie Eingangs bereits angedeutet, werden wir nun unsere eigene Version von Math.Pow programmieren. Machen Sie sich dazu Gedanken wie das Potenzieren von Zahlen funktioniert und übertragen sie dies in Quellcode, welcher ausgeführt werden kann.

Bitte beachten Sie, dass es beim Potenzieren von Zahlen sehr schnell zu Problemen bis hin zu unvorhergesehenen Ergebnissen kommen kann. Machen Sie sich deshalb am besten vorher Gedanken welche Probleme auftreten und wie Sie diese umgehen können.

Für den Extrakick an Nervenkitzel können Sie Ihr Programm Parametrisieren, um die zu berechnenden Zahlen direkt als Konsolenaufruf zu übergeben. Wer es noch einen Tick schwerer will, kann eine Menüführung implementieren, welche die zu berechnenden Zahlen abfragt.

4. Primzahlenprüfung

Als letzte Teilaufgabe dieser Übung geht es um das Implementieren einer eigenen Primzahlenprüfung. Gehen Sie dazu wie folgt vor:

- Erzeugen Sie zunächst ein neues Package mit der Bezeichnung math und kopieren Sie anschließend die Datei MyMath.java in das erzeugte Package
- Vervollständigen Sie in der Klasse MyMath.java die Methode mit der Signatur public static boolean isPrime(BigInteger n) in Zeile 136
- Erinnern Sie sich an die Lerninhalte in dieser Übung (java.math.BigInteger & java.lang.Comparable). Hinweis: Sie können die Methode in Zeile 109 als „Vorlage“ verwenden, ein einfaches Kopieren wird allerdings nicht funktionieren.
- Zum Abschluss überlegen Sie sich eine geeignete Art und Weise wie Sie automatisch die nächste Zahl prüfen lassen bis eine Primzahl gefunden wurde.
- Wie bezeichnet man Funktionen mit gleichem Namen aber unterschiedlichen Signaturen:

  _____

Hinweis: Die Datei MyMath.java befindet sich im Ordner Übung_01_Vorlage.zip