Lehrziele:

- Implementierung von Methoden nach vorgegebenen verbalen Schnittstellenbeschreibungen
- Vertiefung String-, Character- und Integerdatentypen

Aufgabenstellung

In dieser Aufgabe sollen Sie ein C#-Programm schreiben, welches beliebig große natürliche Zahlen addieren und multiplizieren kann.

Das Programm soll/muss aus folgenden Methoden bestehen:

ReadBigInteger

Diese Methode liest einen beliebig langen Text ein, und liefert diesen als String zurück. Solange die Eingabe andere Zeichen als Ziffern enthält, wird die Eingabe wiederholt.

- Parameter:
  - Ein String-Parameter gibt an, welche Eingabeaufforderung dem Benutzer vor dem Einlesen des Textes angezeigt werden soll.
- Rückgabewert:
  - Ein String, der nur Ziffern enthält

IsValidBigInteger

Diese Methode überprüft den übergebenen String auf eine gültige Zahl.

- Parameter:
  - Der String, der überprüft werden soll.
- Rückgabewert:
  - `True` wenn als Parameter ein String übergeben wurde, der nur Ziffern enthält.

AddBigInteger

Diese Methode soll zwei BigInteger-Strings ziffernweise addieren und das Ergebnis als String zurückliefern.

- Parameter:
  - Ein String für den ersten Operanden
  - Ein String für den zweiten Operanden
- Rückgabewert:
  - Die Summe als BigInteger-String

Lösungsansatz:

- Von der niedrigsten bis zur höchsten Stelle der Strings:
  - Jedes einzelne Zeichen auf eine Ziffer umwandeln.
  - Die beiden Ziffern addieren und dabei einen allfälligen Übertrag aus der vorherigen Stelle berücksichtigen.

    - Die Einerstelle in den Ergebnisstring schreiben und die Zehnerstelle als Übertrag für die nächste Stelle aufheben.
- Achten Sie darauf, dass der Übertrag auch an der höchsten Stelle richtig berücksichtigt wird!
- Achten sie auch darauf, dass die Strings ungleich lange sein können.

MultiplyBigIntegerWithDigit

Diese Methode führt eine Multiplikation einer BigInteger-Zahl mit einer einzelnen Ziffer zwischen 0 und 9 durch.

- Parameter:
    - Ein String für den BigInteger, der multipliziert werden soll
    - Eine Zahl zwischen 0 und 9, mit der der BigInteger multipliziert werden soll.
- Rückgabewert:
    - Der String mit dem Rechenergebnis

Lösungsansatz:

- Jede einzelne Stelle im Multiplikand ist mit dem Multiplikator zu multiplizieren und ein allfälliger Übertrag bei der Multiplikation der nächsten Stelle zu berücksichtigen.
- Achten Sie darauf, dass auch der Übertrag an der höchsten Stelle richtig berücksichtigt wird!

MultiplyBigIntegers

Diese Methode führt eine Multiplikation zweier BigInteger-Zahlen durch und liefert das Produkt der beiden Zahlen als BigInteger-String zurück.

- Parameter:
    - Ein String für den BigInteger-Multiplikand
    - Ein String für den BigInteger-Multiplikator
- Rückgabewert:
    - Produkt als BigInteger-String

Lösungsansatz:

- Der Multiplikand muss mit jeder einzelnen Stelle im Multiplikator multipliziert werden.
- Verwenden Sie dazu die Methode ***MultiplyBigIntegerWithDigit***!
- Jedes Zwischenprodukt muss mithilfe der Methode AddBigIntegers zum bisherigen Gesamtergebnis addiert werden. Achten Sie dabei darauf, dass sich mit jeder Stelle die Wertigkeit des Gesamtergebnisses um eine 10er-Stelle erhöht!

Die oben angeführten Methoden müssen wie beschrieben umgesetzt werden. Es ist darauf zu achten, dass die Methodennamen, die Parameter und auch die Rückgabewerte der Beschreibung entsprechen.

Zusätzlich zu den Methoden ist ein Programm zu schreiben, welches die Methoden verwendet. Es sollen dabei zwei (bigInteger) Zahlen eingelesen werden und anschließend ist die Summe und das Produkt auszugeben. Danach wird der Benutzer gefragt, ob eine weitere Eingabe/Berechnung durchgeführt werden soll.

Bildschirmausgabe

```
Big integer
***********
Please enter 1. number: 999
Please enter 2. number: 12

Summe:   1011
Product: 11988

Continue with "y": y
Please enter 1. number: 12345678901234567890123456789O
Please enter 2. number: 9876543210

Summe:   12345678901234567911111111100
Product: 121932631124828532112482853211126352690O

Continue with "y":
```

Zusatzaufgabe

- Erlauben sie auch negative Zahlen

Hintergrund

Wie Sie der Microsoft-Dokumentation zu den in C# unterstützten einfachen Datentypen (https://msdn.microsoft.com/de-de/library/cs7y5x0x%28v=vs.90%29.aspx) entnehmen können, stehen in C# verschiedene Datentypen zum Rechnen mit positiven und negativen ganzen Zahlen zur Verfügung.

Der größte dieser ganzzahligen Datentypen ist „`decimal`“ mit einer Speichergröße von 128 Bits oder 16 Bytes. Das ist der Wertebereich von `decimal`:

```
-79228162514264337593543950335 .. 79228162514264337593543950335
```

Das Rechnen mit noch größeren Zahlen ist nur möglich, wenn man dazu auf Zeichenketten zum Abspeichern der Zahlen ausweicht und die Berechnungen für jede einzelne Stelle der Zeichenkette selbst durchführt.