

## Einschränkungen

- Sie dürfen **keine** zusätzlichen eigenen Hilfsmethoden oder Klassenvariablen verwenden.
- Die vorgegebenen Methodenköpfe dürfen **nicht** erweitert oder geändert werden.
- Sie dürfen Strings **nicht** per Referenz vergleichen.
- Sie dürfen **nicht** die Klassen `StringBuffer` bzw. `StringBuilder` verwenden.
- Sie dürfen **keine** Lambdas, Streams oder Methodenreferenzen verwenden.
- Sie dürfen **nur** folgende Methode(n) aus der Klasse `String` verwenden: `charAt`, `equals`, `length`, `substring`.
- Aus der Klasse `Math` dürfen alle Methoden verwendet werden. Sonst darf **keine** weitere Klasse verwendet werden.

## Aufgabenstellung

Implementieren Sie nur folgende statische Methoden:

- `int myMult(int x, int y)` multipliziert `x` mit `y` ohne den Operator `*` zu benutzen und gibt das Ergebnis zurück.  
*Annahme(n):* `x > 0, y > 0`.
- `char findMaxChar(String text)` sucht das Zeichen mit dem größten ASCII-Wert innerhalb des Strings `text` und gibt dieses zurück.  
*Annahme(n):* `text.length() > 0`.
- `String replaceNthChar(String text, int n, char replaceChar)` ersetzt jedes Zeichen an einer Position, die ein echtes Vielfaches von `n` ist (also `n`, `n*2`, `n*3`, usw.) im String `text` durch das Zeichen `replaceChar` und gibt diesen String zurück.  
*Annahme(n):* `text.length() > 0, n > 0`.
- `void printPattern(int n, char character)` gibt auf der Konsole `n` Zeilen aus. In der ersten Zeile wird das Zeichen `character` `n`-mal ausgegeben. Ab der zweiten Zeile wird die Ausgabe immer um jeweils eine Position nach rechts verschoben und es wird immer ein Zeichen `character` mehr als in der Zeile davor ausgegeben (also `n + 1` Zeichen in Zeile 2, `n + 2` Zeichen in Zeile 3 usw.).  
*Annahme(n):* `n > 0`.

Implementieren Sie folgende Punkte in `main`:

- Deklarieren Sie eine short-Variable `result` und eine String-Variable `test` mit dem Inhalt `"Teststring_Einstufungstest"`.
- `result` soll das Ergebnis des Aufrufs `myMult(3, 7)` ohne Compiler-Fehler zugewiesen werden.

Testen Sie alle Methoden in `main` mit zumindest folgenden Aufrufen, und erzeugen Sie mit deren Hilfe die gezeigten Ausgaben. **Außer `printPattern` darf keine der implementierten Methoden eine Ausgabe erzeugen. Alle anderen Ausgaben müssen in `main` erfolgen.**

Aufruf	Ausgabe in main auf der Konsole
<code>myMult(21, 4)</code>	84
<code>myMult(2, 3500)</code>	7000
<code>findMaxChar(test)</code>	u
<code>findMaxChar("abcdefg")</code>	g
<code>findMaxChar("4321")</code>	4
<code>replaceNthChar(test, 1, '0')</code>	T000000000000000000000000000000
<code>replaceNthChar(test, 3, '#')</code>	Tes#st#in#_E#ns#uf#ng#te#t
<code>replaceNthChar(test, 10, '-')</code>	Teststring-Einstufun-stest
<code>replaceNthChar(test, 30, 'X')</code>	Teststring_Einstufungstest
<code>replaceNthChar("A", 1, '#')</code>	A
<code>printPattern(3, '*')</code>	*** **** *****
<code>printPattern(5, '+')</code>	+++++ +++++ +++++++ +++++++ +++++++

Methode	Bewertungsgrundlage	Punkt(e)
myMult - 5 Punkte	Schleife	2
	Korrekte Berechnung	2
	Rückgabe	1
findMaxChar - 4 Punkte	Schleife	2
	Verzweigung	1
	Rückgabe	1
replaceNthChar - 6 Punkte	Schleife	2
	Verzweigung	2
	Randfälle (Position 0) korrekt behandelt	1
	Rückgabe	1
printPattern - 6 Punkte	Richtige Anzahl an Zeilen	2
	Richtige Anzahl an Zeichen pro Zeile	2
	Richtiger Versatz	2
main - 4 Punkte	Deklarationen	1
	Korrekte Zuweisung (result)	1
	Korrekte Aufrufe	1
	Ausgaben in der Konsole	1
Gesamt		25

## Einschränkungen

- Sie dürfen **keine** zusätzlichen eigenen Hilfsmethoden oder Klassenvariablen verwenden.
- Die vorgegebenen Methodenköpfe dürfen **nicht** erweitert oder geändert werden.
- Sie dürfen Strings **nicht** per Referenz vergleichen.
- Sie dürfen **nicht** die Klassen `StringBuffer` bzw. `StringBuilder` verwenden.
- Sie dürfen **keine** Lambdas, Streams oder Methodenreferenzen verwenden.
- Sie dürfen **nur** folgende Methode(n) aus der Klasse `String` verwenden: `charAt`, `equals`, `length`, `substring`.
- Aus der Klasse `Math` dürfen alle Methoden verwendet werden. Sonst darf **keine** weitere Klasse verwendet werden.

## Aufgabenstellung

Implementieren Sie nur folgende statische Methoden:

- `int countDivisors(int x, int y)` liefert die Anzahl der Zahlen im Intervall  $[x, y]$  zurück, die durch 4 teilbar sind, aber *nicht* durch 6.  
*Annahme(n):*  $x > 0, y > 0, x \leq y$ .
- `int findDoubles(String text)` zählt, wie oft der Buchstabe 'a' innerhalb des Strings `text` zweimal direkt hintereinander vorkommt, und liefert diesen Wert zurück.  
*Annahme(n):* `text.length() > 0`.
- `String reverseInsert(String text, char character)` fügt zwischen je zwei Zeichen von `text` den Buchstaben `character` ein und gibt diesen String in umgekehrter Reihenfolge zurück.  
*Annahme(n):* `text.length() > 1`.
- `void printPattern(int n, char character)` gibt auf der Konsole `n` Zeilen aus. Ist die aktuelle Zeilennummer gerade, wird 2·`n`-mal das Zeichen `character` ausgegeben. Ist die aktuelle Zeilennummer ungerade, wird `n`-mal zuerst das Zeichen `character` gefolgt von einem Punkt '.' ausgegeben. Am Ende jeder Zeile wird die Zeilennummer ausgegeben. Die Zählung der Zeilennummern beginnt mit 1.  
*Annahme(n):*  $n > 0$ .

Implementieren Sie folgende Punkte in `main`:

- Deklarieren Sie eine short-Variable `result` und eine String-Variable `test` mit dem Inhalt `"Teststring_Einstufungstest"`.
- `result` soll das Ergebnis des Aufrufs `countDivisors(299, 305)` ohne Compiler-Fehler zugewiesen werden.

Testen Sie alle Methoden in `main` mit zumindest folgenden Aufrufen, und erzeugen Sie mit deren Hilfe die gezeigten Ausgaben. **Außer `printPattern` darf keine der implementierten Methoden eine Ausgabe erzeugen. Alle anderen Ausgaben müssen in `main` erfolgen.**

Aufruf	Ausgabe in main auf der Konsole
<code>countDivisors(1, 28)</code>	5
<code>countDivisors(101, 2001)</code>	317
<code>countDivisors(8, 8)</code>	1
<code>findDoubles(test)</code>	0
<code>findDoubles("Haarspangenaal")</code>	2
<code>findDoubles("The Black Beast of Aaaaargh!")</code>	3
<code>findDoubles("Schokoladenkuchen")</code>	0
<code>reverseInsert(test, '.')</code>	t.s.e.t.s.g.n.u.f.u.t.s.n.i.E._.g.n.i.r.t.s.t.s.e.T
<code>reverseInsert("qwerty", '-')</code>	y-t-r-e-w-q
<code>reverseInsert("Pinkie Pie", '!')</code>	e!i!P! !e!i!k!n!i!P
<code>printPattern(4, '!')</code>	!.!.!.!.1 !!!!!!!2 !.!.!.!.3 !!!!!!!4
<code>printPattern(5, '*')</code>	*.*.*.*.1 *****2 *.*.*.*.3 *****4 *.*.*.*.5

Methode	Bewertungsgrundlage	Punkt(e)
countDivisors - 5 Punkte	Schleife	2
	Korrekte Berechnung	2
	Rückgabe	1
findDoubles - 4 Punkte	Schleife	2
	Verzweigung	1
	Rückgabe	1
reverseInsert - 6 Punkte	Schleife(n)	2
	Verzweigung	2
	Randfälle (Position text.length()) korrekt behandelt	1
	Rückgabe	1
printPattern - 6 Punkte	Richtige Anzahl an Zeilen	2
	Richtige Anzahl an Zeichen pro Zeile	2
	Korrekte Zeilennummern	2
main - 4 Punkte	Deklarationen	1
	Korrekte Zuweisung (result)	1
	Korrekte Aufrufe	1
	Ausgaben in der Konsole	1
Gesamt		25

## Einschränkungen

- Sie dürfen **keine** zusätzlichen eigenen Hilfsmethoden oder Klassenvariablen verwenden.
- Die vorgegebenen Methodenköpfe dürfen **nicht** erweitert oder geändert werden.
- Sie dürfen Strings **nicht** per Referenz vergleichen.
- Sie dürfen **nicht** die Klassen `StringBuffer` bzw. `StringBuilder` verwenden.
- Sie dürfen **keine** Lambdas, Streams oder Methodenreferenzen verwenden.
- Sie dürfen **nur** folgende Methode(n) aus der Klasse `String` verwenden: `charAt`, `equals`, `length`, `substring`.
- Sonst darf **keine** weitere Klasse verwendet werden.

## Aufgabenstellung

Implementieren Sie nur folgende **statische** Methoden:

- int getIntegerRoot(int k)** bestimmt, ob `k` das Quadrat einer natürlichen Zahl ist - d.h. ob eine natürliche Zahl `i` existiert, sodass  $i \cdot i = k$  ist. Ist dies der Fall, wird `i` zurückgegeben, ansonsten wird `-1` zurückgegeben.  
*Annahme(n):* `k > 0`.  
*Hinweis:* Die Berechnung muss nicht effizient implementiert werden. Bedenken Sie, dass für `i` nur Zahlen im Intervall `[1,k]` in Frage kommen. Beachten Sie auch, dass Sie gemäß den Einschränkungen keine Methoden der Klasse `Math` verwenden dürfen.
- String getThird(String a, String b, String c)** vergleicht den Inhalt der Parameter `a`, `b` und `c` und gibt davon abhängig einen String zurück. Sind alle drei Parameter gleich, ist der Rückgabewert **"alle gleich"**. Sind alle drei Parameter unterschiedlich, ist der Rückgabewert **"alle unterschiedlich"**. Sind genau zwei Parameter gleich, wird der dritte (zu den anderen unterschiedliche) zurück gegeben.  
*Annahme(n):* `a != null, b != null, c != null`
- String replaceA(String s)** gibt eine Kopie von `s` zurück, bei der das erste Vorkommen des Zeichens `'a'` durch die Zahl `1`, das zweite Vorkommen von `'a'` durch die Zahl `2` usw. ersetzt wird.  
*Annahme(n):* `s != null`
- void printBars(int i)** gibt die Werte im Intervall `[1,i]` aufsteigend in zwei Zeilen aus. Die erste Zeile enthält alle Werte, die nicht durch `3` teilbar sind. Die zweite Zeile enthält alle Werte, die durch `3` teilbar sind. Nach jedem Wert wird abwechselnd das Zeichen `'-'` bzw. `'+'` ausgegeben. Die Trennzeichen der ersten Zeile beginnen mit `'-'`, die der zweiten Zeile mit `'+'`.

Implementieren Sie folgende Punkte in `main`:

- Deklarieren Sie eine short-Variable `result` und eine String-Variable `test` mit dem Inhalt **"Blaukraut"**.
- `result` soll das Ergebnis des Aufrufs `getIntegerRoot(25)` ohne Compiler-Fehler zugewiesen werden.

Testen Sie alle Methoden in `main` mit zumindest folgenden Aufrufen, und erzeugen Sie mit deren Hilfe die gezeigten Ausgaben. **Außer `printBars` darf keine der implementierten Methoden eine Ausgabe erzeugen. Alle anderen Ausgaben müssen in `main` erfolgen.**

Aufruf	Ausgabe in main auf der Konsole
<code>getIntegerRoot(144)</code>	12
<code>getIntegerRoot(13)</code>	-1
<code>getIntegerRoot(1)</code>	1
<code>getThird("toss", "a", "coin")</code>	alle unterschiedlich
<code>getThird("Blaukraut", "bleibt", test)</code>	bleibt
<code>getThird("badger", "badger", "badger")</code>	alle gleich
<code>replaceA("TU Wien")</code>	TU Wien
<code>replaceA("Hubba bubba!")</code>	Hubb1 bubb2!
<code>replaceA("aaaa")</code>	1234
<code>printBars(2)</code>	1-2+
<code>printBars(19)</code>	1-2+4-5+7-8+10-11+13-14+16-17+19- 3+6-9+12-15+18-
<code>printBars(20)</code>	1-2+4-5+7-8+10-11+13-14+16-17+19-20+ 3+6-9+12-15+18-
<code>printBars(21)</code>	1-2+4-5+7-8+10-11+13-14+16-17+19-20+ 3+6-9+12-15+18-21+

Methode	Bewertungsgrundlage	Punkt(e)
getIntegerRoot - 5 Punkte	Schleife	2
	Korrekte Berechnung	2
	Rückgabe	1
getThird - 6 Punkte	Korrekter Vergleich	2
	Verzweigungen	2
	Rückgabe	2
replaceA - 5 Punkte	Schleife	2
	Verzweigung	2
	Rückgabe	1
printBars - 5 Punkte	Richtige Anzahl an Zeilen	1
	Richtige Anzahl an Zeichen pro Zeile	2
	Korrekter Zeileninhalt	2
main - 4 Punkte	Deklarationen	1
	Korrekte Zuweisung (result)	1
	Korrekte Aufrufe	1
	Ausgaben in der Konsole	1
Gesamt		25