Loops

Problem

Prompt the user for a positive integer

```
    Enter a positive integer:
```

so that the integer n is entered by the user on the same line as the prompts. Print a line with 1 $*$, a line with 2 $*$'s, $\ldots$, a line with $n-1$ $*$'s, and a line with n $*$'s.

Answer

```
output.print("Enter a positive integer: ");
int n = input.nextInt();
for (int r = 0; r < n; r++)
{
   for (int c = 0; c <= r; c++)
   {
      output.print("*");
   }
   output.println();
}
```

Loops

Problem

Prompt the user for a positive integer

```
    Enter a positive integer:
```

so that the integer n is entered by the user on the same line as the prompts. Print a line with 1 $*$, a line with 2 $*$'s, $\ldots$, a line with $n-1$ $*$'s, and a line with n $*$'s. Reprompt the user if they enter a non-positive integer.

Answer

```
int n;
do
{
   output.print("Enter a positive integer: ");
   n = input.nextInt();
}
while (n <= 0);

for (int r = 0; r < n; r++)
{
   for (int c = 0; c <= r; c++)
   {
         output.print("*");
      }
      output.println();
   }
```

Check03A

Number of students enrolled in the course: 225

Number of students that eChecked Check03A: 70 (31%)

Check04D

Number of students enrolled in the course: 215

Number of students that eChecked Check04D: 46 (21%)

Check05C

Number of students enrolled in the course: 212

Number of students that eChecked Check05C: 38 (18%)

Strings and Loops

CSE 1020

October 20, 2010

Strings

Strings are immutable objects.

The state of an immutable object cannot be changed.

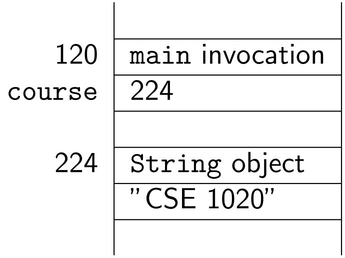
The `String` API does not contain any mutators.

The `StringBuffer` class provides mutable strings. [1]

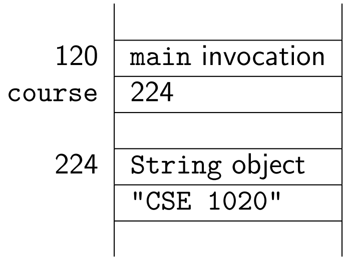
[1] We will come back to the `StringBuffer` class later in this lecture.

Strings

```
String course = new String("CSE 1020");
```



Strings



String reference: `course`
String object
String literal: `"CSE 1020"`

Strings are Everywhere

Instead of

```
String course = new String("CSE 1020");
```

we are allowed to write

```
String course = "CSE 1020";
```

Although in most cases you may think of `"CSE 1020"` and `new String("CSE 1020")` as synonyms, they are not always equivalent.[2]

[2] Hardly ever will this difference impact your app.

Strings are Immutable

According to the Java Language Specification,

> *Strings that are the values of constant expressions are "interned" so as to share unique instances*

James Gosling, Bill Joy, Guy L. Steele Jr. and Gilad Bracha. The Java Language Specification. Third edition. Addison-Wesley.2005.

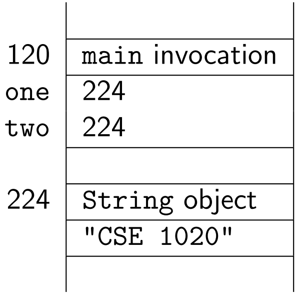
Strings are Immutable

Strings that are the values of constant expressions are "interned" so as to share unique instances

These constant expressions are built from String literals and the binary operator $+$.

Strings are Immutable

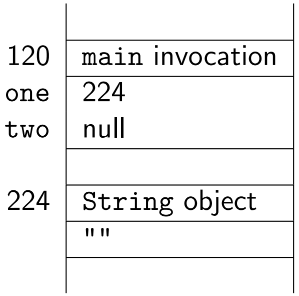
```
String one = "CSE 1020";
String two = "CSE" + " " + "1020";
```



This saves memory. Why can `one` and `two` refer to the same `String` object?

The Empty String versus null

```
String one = "";
String two = null;
```



My Mom . . .

. . . tends to write long sentences. Long sentences are in general more difficult to comprehend.

Rudolf Flesch. A new readability yardstick. *Journal of Applied Psychology*, 32(3): 221-233, June 1948.

Problem 6_1

Problem

Prompt the user for a file name by printing

```
    Enter file name:
```

so that the file name is entered by the user on the same line as the prompt.

You may assume that the file consists of sentences.

Print the last word of every sentence, each on a separate line.

Problem 6_2

Problem

Prompt the user for a file name by printing

```
    Enter file name:
```

so that the file name is entered by the user on the same line as the prompt.

You may assume that the file consists of sentences.

Print the number of words for every sentence, each on a separate line.

Problem 6_3

Problem

Prompt the user for a file name by printing

```
    Enter file name:
```

so that the file name is entered by the user on the same line as the prompt.

You may assume that the file consists of sentences.

Print those sentences that have more than 35 words, each on a separate line.[a]

[a]The New Yorker of October 26, 1946 has on average 20 words per sentence.

The `StringBuffer` Class

StringBuffers are mutable objects.

The method

```
    public StringBuffer append(String s)
```

adds the String `s` to the end of the StringBuffer.

The method returns a reference to the StringBuffer itself. Although this is not needed (why?), it is convenient.

From String to StringBuffer and Back

$$\begin{tikzcd}\texttt{String} \arrow[r, bend left, "\texttt{new StringBuffer(}\cdot\texttt{)}"] & \texttt{StringBuffer} \arrow[l, bend left, "\cdot\texttt{.toString()}"]\end{tikzcd}$$

Problem 6_4

Problem

The file name is provided as a command-line argument.

You may assume that the file consists of sentences.

Print those sentences that have more than 35 words, each on a separate line.

If the user does not provide a command-line argument, print

```
    Use:  java Problem6_4 <file name>
```

Command-Line Arguments

Question

How does the user provide command-line arguments?

Command-Line Arguments

Question

How does the user provide command-line arguments?

Answer

```
java Problem6_4 "book.txt"
```

Command-Line Arguments

Question

How does the user provide command-line arguments?

Answer

```
java Problem6_4 "book.txt"
```

Question

How does the client get the command-line arguments?

Command-Line Arguments

Question

How does the user provide command-line arguments?

Answer

```
java Problem6_4 "book.txt"
```

Question

How does the client get the command-line arguments?

Answer

As the parameter of the `main` method.

Command-Line Arguments

Question

How does the user provide command-line arguments?

Answer

```
java Problem6_4 "book.txt"
```

Question

How does the client get the command-line arguments?

Answer

As the parameter of the `main` method.

Question

What is the type of the parameter of the `main` method.

Command-Line Arguments

Question

How does the user provide command-line arguments?

Answer

```
java Problem6_4 "book.txt"
```

Question

How does the client get the command-line arguments?

Answer

As the parameter of the `main` method.

Question

What is the type of the parameter of the `main` method.

Answer

`String[]`: an array of `String`s.

Command-Line Arguments

```
public static void main(String[] arguments)
```

Question

How does the client get the first command-line arguments?

Command-Line Arguments

```
public static void main(String[] arguments)
```

Question

How does the client get the first command-line arguments?

Answer

`arguments[0]`

Command-Line Arguments

```
public static void main(String[] arguments)
```

Question

How does the client get the first command-line arguments?

Answer

`arguments[0]`

Question

How does the client get the second command-line arguments?

Command-Line Arguments

```
public static void main(String[] arguments)
```

Question

How does the client get the first command-line arguments?

Answer

`arguments[0]`

Question

How does the client get the second command-line arguments?

Answer

`arguments[1]`

Command-Line Arguments

```
public static void main(String[] arguments)
```

Question

How does the client get the number of command-line arguments?

Command-Line Arguments

```
public static void main(String[] arguments)
```

Question

How does the client get the number of command-line arguments?

Answer

`arguments.length`

Problem 6_4

Problem

The file name is provided as a command-line argument.

You may assume that the file consists of sentences.

Print those sentences that have more than 35 words, each on a separate line.

If the user does not provide a command-line argument, print

```
    Use:  java Problem6_4 <file name>
```

Problem 6_5

Problem

The file name is provided as a command-line argument.

You may assume that the file consists of sentences.

Replace each sequence of 15 or 16 digits with 15 or 16 *'s.

If the user does not provide a command-line argument, print

```
    Use:  java Problem6_5 <file name>
```

Regular Expressions

A regular expression allows us to express a pattern. For example, the pattern 15 digits can be expressed by the regular expression

$$\backslash d\{15\}?$$

The pattern $\backslash d$ matches any digit and the pattern $p\{n\}?$, where p is a pattern and n is a natural number, matches p exactly n times.

A detailed description of patterns can be found in the API of the `Pattern` class, which is part of the `java.util.regex` package.

Regular Expressions

The class `String` contains the following methods.

```
public boolean matches(String regex)
```

tests whether this string matches the given regular expression.

```
public String replaceAll(String regex, String
replacement)
```

replaces each substring of this string that matches the given regular expression with the given replacement.

Study regular expressions. There will be a question about them on next week's test.

Problem 6_6

Problem

The distribution file name is provided as the first command-line argument. This file contains for each student, their CSE number, their York email account, their student number, and their name (last name, first name), separated by tabs.

The CSE number is provided as the second command-line argument.

Print

```
    To: york@yorku.ca
    Dear name,
```

where *york* is their York email account and *name* is their name (first name followed by last name).

If the user does not provide two command-line arguments, print

```
    Use:  java Problem6_6 <file name> <cse number>
```

Regular Expressions

The class String contains the following method.

```
    public String[] split(String regex)
```

splits this string around matches of the given regular expression.

Problem 6_7

Problem

If the use does not provide a command-line argument, print

```
    Use:  java Problem 6_7 USD<amount>
```

If the user does not provide a command-line argument of the form USDa, where a is a non-negative real number, print

```
    Use:  java Problem 6_7 USD<amount>
```

If the user provides a command-line argument of the form USDa, where a is a non-negative real number, print a.

From String to double and Back

$$\begin{tikzcd}
\texttt{String} \arrow[rr, bend left, "\texttt{Double.parseDouble(}\cdot\texttt{)}"] & \texttt{Double} \arrow[l, bend left, "\cdot\texttt{.toString()}"] & \texttt{double} \arrow[l, bend left, "\texttt{new Double(}\cdot\texttt{)}"]
\end{tikzcd}$$