CSE 214

Output: The format method

Jeremy Morris
Sometimes we want nicely formatted output
  “print” and “println” are cumbersome for printing multiple variables on a line

Consider this line of output:
  The box has 12 red roses and 8 white roses
  How would we print that to the screen?
Consider this line of output:

- The box has 12 red roses and 8 white roses
- How would we print that to the screen?

Using print and println:

```java
int [] box = new array[2];
box[0]=12;
box[1]=8;
System.out.print("The box has ");
System.out.print(box[0]);
System.out.print(" red roses and ");
System.out.print(box[1]);
System.out.println(" white roses");
```
Formatted output

Consider this line of output:

- The box has 12 red roses and 8 white roses
- How would we print that to the screen?

All in one statement:

```java
int [] box = new array[2];
box[0]=12;
box[1]=8;
System.out.println("The box has "+box[0]+" red roses and "+box[1]+" white roses");
```
Consider this line of output:
- The box has 12 red roses and 8 white roses
- How would we print that to the screen?

Using format:
```java
int[] box = new array[2];
box[0]=12;
box[1]=8;
System.out.format("The box has %d red roses and %d white roses",box[0],box[1]);
System.out.println();
```
format method

The format method takes at least two arguments

- A format string ("how to print")
- A list of variables ("what to print")

The format string is a combination of typical string output and format specifiers

"The box has %d red roses and %d white roses"

- Format specifiers always start with a ‘%’ and end with a character telling the type of the variable to be printed
Format specifiers

- Typical format specifiers:
  - %d – “decimal integer” – i.e. base 10 int
  - %f – “floating point number” – i.e. double, float
  - %s – “string” – i.e. String
Format specifier elements

- Between the ‘%’ and the type character, a number of elements can be set
  - Some important ones are the width, precision and padding flags

- The width element
  - %20s, %5d, %4f
  - Sets the minimum width of the formatted value in characters
  - If the value being printed is not this wide, the value is left-padded with spaces
The width element

String value = "Test123";
System.out.format("The value is %s",value);

**Output:** The value is Test123

System.out.format(""The value is %10s",value);

**Output:** The value is Test123
The precision element

- `%5.2f`
  - For floating point, this sets the number of places after the decimal that will be printed
    - The above prints exactly 2 digits after the decimal

- `%5.20s`
  - For strings, this sets the maximum length of a string
    - The above prints a string with a minimum width of 5 (padded if needed) and a maximum width of 20 (truncated if necessary)
The precision element

```java
String value = "Test123";
System.out.format("The value is %s",value);

Output: The value is Test123

System.out.format("The value is %.5s",value);

Output: The value is Test1
```
The padding flag

%-5.20s

- By default, padding is added to the left
- The padding flag (a ‘-’ before the width) forces the padding to be added to the right
The padding flag

String value = "Test123";
System.out.format("Example: %10s is the value", value);

Output: Example: Test123 is the value

String value = "Test123";
System.out.format("Example: %-10s is the value", value);

Output: Example: Test123 is the value
The format method

- **format** allows you to create well-formatted data
  - For example, this code snippet:
    ```java
    System.out.format("%-10.10s %-10.10s %10.2f", firstName, lastName, acctBal);
    ```
  - Can produce output like this:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Marley</td>
<td>55.00</td>
<td></td>
</tr>
<tr>
<td>Joseph</td>
<td>Cool</td>
<td>255.00</td>
<td></td>
</tr>
</tbody>
</table>