MET CS 520 – Information Structures with Java

Input, Output, and Formatting

System.out.println for console output

- System.out is an object that is part of the Java language
- println is a method invoked by the System.out object that can be used for console output
  - The data to be output is given as an argument in parentheses
  - A plus sign is used to connect more than one item
  - Every invocation of println ends a line of output

```
System.out.println("The answer is " + 42);
```

println Versus print

- Another method that can be invoked by the System.out object is print
- The print method is like println, except that it does not end a line
  - With println, the next output goes on a new line
  - With print, the next output goes on the same line

Formatting Output with printf

- Starting with version 5.0, Java includes a method named printf that can be used to produce output in a specific format
- The Java method printf is similar to the print method
  - Like print, printf does not advance the output to the next line
- System.out.printf can have any number of arguments
  - The first argument is always a format string that contains one or more format specifiers for the remaining arguments
  - All the arguments except the first are values to be output to the screen

printf Format Specifier

- The code

```
double price = 19.8;
System.out.print("$");
System.out.printf("%6.2f", price);
System.out.println(" each");
```

- will output the line
  - $ 19.80 each
- The format string "%6.2f" indicates the following:
  - End any text to be output and start the format specifier (%)
  - Display up to 6 right-justified characters, pad fewer than six characters on the left with blank spaces (i.e., field width is 6)
  - Display exactly 2 digits after the decimal point (.2)
  - Display a floating point number, and end the format specifier (i.e., the conversion character is f)

Right and Left Justification in printf

- The code

```
double value = 12.123;
System.out.printf("Start%8.2fEnd", value);
System.out.println();
System.out.printf("Start%-8.2fEnd", value);
System.out.println();
```

  - will output the following

```
Start   12.12End
Start12.12   End
```

- The format string "Start%8.2fEnd" produces output that is right justified with three blank spaces before the 12.12
- The format string "Start%-8.2fEnd" produces output that is left justified with three blank spaces after the 12.12

Multiple arguments with printf

- The following code contains a printf statement having three arguments
  - The code

```
double price = 19.8;
String name = "magic apple";
System.out.printf("$%6.2f for each %s.", price, name);
System.out.println();
System.out.println("Wow");
```

  - will output

```
$ 19.80 for each magic apple.
Wow
```

  - Note that the first argument is a format string containing two format specifiers (%6.2f and %s)
  - These format specifiers match up with the two arguments that follow (price and name)

Line Breaks with printf

- Line breaks can be included in a format string using %n
- The code

```
double price = 19.8;
String name = "magic apple";
System.outprintf("$%6.2f for each %s.%n", price,
name);
System.out.println("Wow");
```

- will output

```
$ 19.80 for each magic apple.
Wow
```

Format Specifiers for System.out.printf

Display 2.1 Format Specifiers for `System.out.printf`

CONVERSION CHARACTER	TYPE OF OUTPUT	EXAMPLES
d	Decimal (ordinary) integer	%5d %d
f	Fixed-point (everyday notation) floating point	%6.2f %f
e	E-notation floating point	%8.3e %e
g	General floating point (Java decides whether to use E-notation or not)	%8.3g %g
s	String	%12s %s
c	Character	%2c %c

Demo

PrintfDemo.java

Formatting Money Amounts with printf

- A good format specifier for outputting an amount of money stored as a double type is %.2f
- It says to include exactly two digits after the decimal point and to use the smallest field width that the value will fit into:

```
double price = 19.99;
System.out.printf("The price is $%.2f each.")
```

- produces the output:

```
The price is $19.99 each.
```

Legacy Code

- Code that is "old fashioned" but too expensive to replace is called legacy code
- Sometimes legacy code is translated into a more modern language
- The Java method printf is just like a C language function of the same name
- This was done intentionally to make it easier to translate C code into Java

Money Formats

- Using the NumberFormat class enables a program to output amounts of money using the appropriate format
  - The NumberFormat class must first be imported in order to use it

    ```
    import java.text.NumberFormat
    ```
  - An object of NumberFormat must then be created using the getCurrencyInstance() method
  - The format method takes a floating-point number as an argument and returns a String value representation of the number in the local currency

Demo

CurrencyFormatDemo.java

Locale Constants for Currencies of Different Countries

Display 2.4 Locale Constants for Currencies of Different Countries

`Locale.CANADA`	Canada (for currency, the format is the same as US)
`Locale.CHINA`	China
`Locale.FRANCE`	France
`Locale.GERMANY`	Germany
`Locale.ITALY`	Italy
`Locale.JAPAN`	Japan
`Locale.KOREA`	Korea
`Locale.TAIWAN`	Taiwan
`Locale.UK`	United Kingdom (English pound)
`Locale.US`	United States

Importing Packages and Classes

- Libraries in Java are called packages
  - A package is a collection of classes that is stored in a manner that makes it easily accessible to any program
  - In order to use a class that belongs to a package, the class must be brought into a program using an import statement
  - Classes found in the package java.lang are imported automatically into every Java program

```
import java.text.NumberFormat;
 // import theNumberFormat class only
import java.text.*;
 //import all the classes in package java.text
```

The DecimalFormat Class

- Using the DecimalFormat class enables a program to format numbers in a variety of ways
  - The DecimalFormat class must first be imported
  - A DecimalFormat object is associated with a pattern when it is created using the new command
  - The object can then be used with the method format to create strings that satisfy the format
  - An object of the class DecimalFormat has a number of different methods that can be used to produce numeral strings in various formats

Demo.java

DecimalFormatDemo.java

Console Input Using the Scanner Class

- Starting with version 5.0, Java includes a class for doing simple keyboard input named the Scanner class
- In order to use the Scanner class, a program must include the following line near the start of the file:

```
import java.util.Scanner
```

- This statement tells Java to
  - Make the Scanner class available to the program
  - Find the Scanner class in a library of classes (i.e., Java package) named java.util

Console Input Using the Scanner Class

- The following line creates an object of the class Scanner and names the object keyboard :

```
Scanner keyboard = new Scanner(System.in);
```

- Although a name like keyboard is often used, a Scanner object can be given any name
  - For example, in the following code the Scanner object is named scannerObject

```
Scanner scannerObject = new Scanner(System.in);
```

- Once a Scanner object has been created, a program can then use that object to perform keyboard input using methods of the Scanner class

Console Input Using the Scanner Class

- The method nextInt reads one int value typed in at the keyboard and assigns it to a variable:

```
int numberOfPods = keyboard.nextInt();
```

- The method nextDouble reads one double value typed in at the keyboard and assigns it to a variable:

```
double d1 = keyboard.nextDouble();
```

- Multiple inputs must be separated by whitespace and read by multiple invocations of the appropriate method
  - Whitespace is any string of characters, such as blank spaces, tabs, and line breaks that print out as white space

Console Input Using the Scanner Class

- The method next reads one string of non-whitespace characters delimited by whitespace characters such as blanks or the beginning or end of a line
- Given the code

```
String word1 = keyboard.next();
String word2 = keyboard.next();
```

- and the input line

```
jelly beans
```

- The value of word1 would be jelly, and the value of word2 would be beans

Console Input Using the Scanner Class

- The method nextLine reads an entire line of keyboard input
  - The following code reads in an entire line and places the string that is read into the variable line

```
String line = keyboard.nextLine();
```

- The end of an input line is indicated by the escape sequence '\n'
  - This is the character input when the Enter key is pressed
  - On the screen it is indicated by the ending of one line and the beginning of the next line
- When nextLine reads a line of text, it reads the '\n' character, so the next reading of input begins on the next line
  - However, the '\n' does not become part of the string value returned (e.g., the string named by the variable line above does not end with the '\n' character)

Demo.java

Scanner.java

Pitfall: Dealing with the Line Terminator, '\n'

- The method nextLine of the class Scanner reads the remainder of a line of text starting wherever the last keyboard reading left off
- This can cause problems when combining it with different methods for reading from the keyboard such as nextInt
- Given the code,

```
Scanner keyboard = new Scanner(System.in);
int n = keyboard.nextInt();
String s1 = keyboard.nextLine();
String s2 = keyboard.nextLine();
```

and the input,

```
2
Heads are better than
1 head.
```

what are the values of n, s1, and s2?

Pitfall: Dealing with the Line Terminator, '\n'

- Given the code and input on the previous slide
  - n will be equal to "2",
  - s1 will be equal to "", and
  - s2 will be equal to "heads are better than"
- If the following results were desired instead
  - n equal to "2",
  - s1 equal to "heads are better than", and
  - s2 equal to "1 head"
- then an extra invocation of nextLine would be needed to get rid of the end of line character ('\n')

Programming Tip: Prompt for Input

- A program should always prompt the user when he or she needs to input some data:

```
System.out.println("Enter the number of pods followed by");
System.out.println("the number of peas in a pod:");
```

Programming Tip: Echo Input

- Always echo all input that a program receives from the keyboard
- In this way a user can check that he or she has entered the input correctly
  - Even though the input is automatically displayed as the user enters it, echoing the input may expose subtle errors (such as entering the letter "O" instead of a zero)

Demo.java

SelfService.java

The Empty String

- A string can have any number of characters, including zero characters
  - "" is the empty string
- When a program executes the nextLine method to read a line of text, and the user types nothing on the line but presses the Enter key, then the nextLine Method reads the empty string

Other Input Delimiters

- The delimiters that separate keyboard input can be changed when using the Scanner class
- For example, the following code could be used to create a Scanner object and change the delimiter from whitespace to "##"

```
Scanner keyboard2 = new Scanner(System.in);
Keyboard2.useDelimiter("##");
```

- After invocation of the useDelimiter method, "##" and not whitespace will be the only input delimiter for the input object keyboard2