

Java Console Input/Output

The Basics

Lecture 3
CGS 3416 Fall 2019

September 3, 2019

Console Output

- ▶ **System.out** – *out* is a *PrintStream* object, a static data member of class *System*. This represents standard output
- ▶ Use this object to call functions *print*, *println*, and even *printf*
 - ▶ **print()** – converts parameter to a string (if not already one) and prints it out
 - ▶ **println()** – prints parameter, and also prints a newline after
 - ▶ **printf** – works like in C programming. Formatted string, followed by parameters to "fill in the blanks"

Sample Calls

```
System.out.print("Hello, World");      // no newline
System.out.println("Hello World");
// adds newline at end
```

```
int feet = 6, inches = 3;
System.out.printf("I am %d feet and %d inches tall\n",
                 feet, inches);
// just like printf in C
```

Concatenation

If the `+` operator is used with at least one string operand, then the operation is string concatenation.

Other types will be auto-converted to type string if needed

```
System.out.println("The number of states in the U.S.  
                    is " + 50);
```

```
int sides = 8;  
System.out.println("Number of sides on a stop  
                    sign = " + sides);
```

Formatting with printf

- ▶ When printing values with decimal precision it is often useful to be able to specify how many decimal places should be printed
- ▶ Format of printf calls:
System.out.printf(format string, list of parameters);
- ▶ The format string is a string in quotes, with special format symbols inserted:
 - ▶ %d specifies an integer
 - ▶ %c specifies a character
 - ▶ %s specifies a String
 - ▶ %f specifies a floating point type
- ▶ Consider the format symbols to be “fill-in-the-blanks” spots in the format string. These are filled in with the list of parameters

printf Example

```
int numStudents = 25;
char letterGrade = 'A';
double gpa = 3.95;

System.out.printf("There are %d students\n",
                  numStudents);
System.out.printf("Bobby's course grade was %c, and
                  his GPA is %f\n", letterGrade, gpa);

// The output from this example is:
//   There are 25 students
//   Bobby's course grade was A, and his GPA is 3.950000
```

printf Example

To specify how many decimal places for the output of a floating point value, modify the '%f' symbol in this format:

`%.Nf` // where N is the number of decimal places

Example:

```
double gpa = 3.275;
```

```
double PI = 3.1415;
```

```
System.out.printf("gpa = %.2f", gpa);
```

```
System.out.printf("PI = %.3f", PI);
```

```
// Output is:
```

```
//    gpa = 3.28
```

```
//    PI = 3.142
```

Console Input

- ▶ Before Java version 1.5.0, console input was harder. Since 1.5.0, we have the `Scanner` class
- ▶ class `Scanner` is a text parser. Contains easy methods for grabbing different types of input
- ▶ `System.in` is an `InputStream` object that represents standard input
- ▶ To use `Scanner` to read from standard input:
 1. Put the appropriate import statement at the top of the file:

```
import java.util.Scanner;
```
 2. Create a `Scanner` object
 3. Pass in `System.in` into the `Scanner` constructor, when creating the object

Example

```
import java.util.Scanner;
// yadda yadda

Scanner input = new Scanner(System.in);

// now we can use the object to read data from
    // the keyboard (stdin).
// Some sample calls:

int x = input.nextInt();
double y = input.nextDouble();
String s = input.next();
```

Reading different types of input

Different data types require the use of different methods available in the `Scanner` class. These are available only for primitive types and `Strings`.

For the following table, assume the `Scanner` object is called "in".

Type	Reading in one variable
<code>int</code>	<code>int i = in.nextInt();</code>
<code>long</code>	<code>long l = in.nextLong();</code>
<code>float</code>	<code>float f = in.nextFloat();</code>
<code>double</code>	<code>double d = in.nextDouble();</code>
<code>boolean</code>	<code>boolean bool = in.nextBoolean();</code>
<code>String</code>	<code>String st = in.next();</code>

Reading in chars

- ▶ The Scanner class does not have a method for reading in characters.
- ▶ However, there are several ways to read in a single character using the Scanner class.
- ▶ One such way is to read in a String and then grab the first character.

```
String st = in.next();  
char c = st.charAt(0);
```