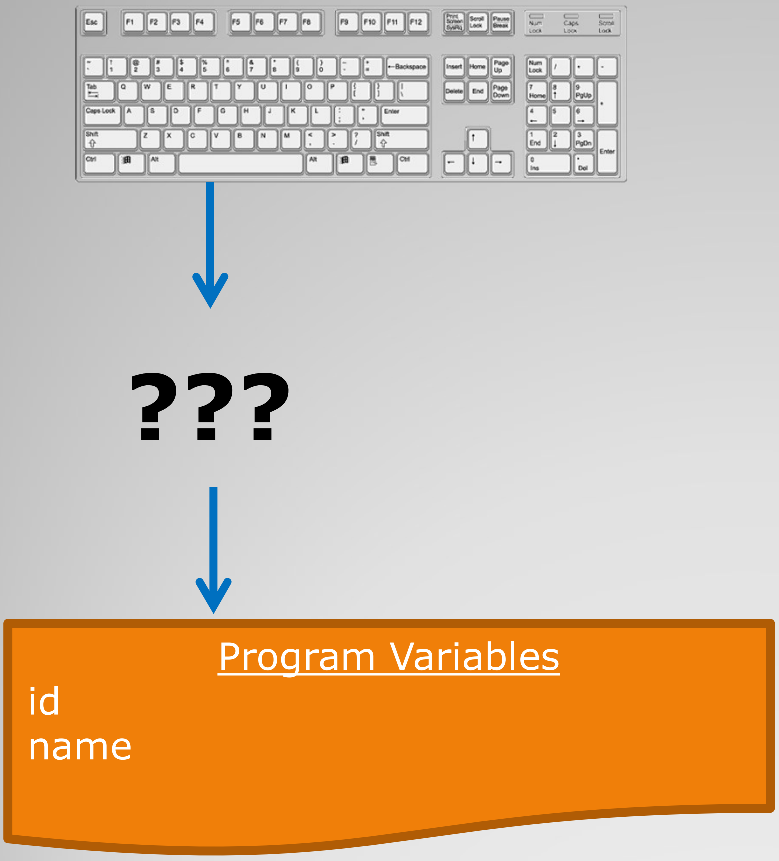
BCS 345 Java Programming

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

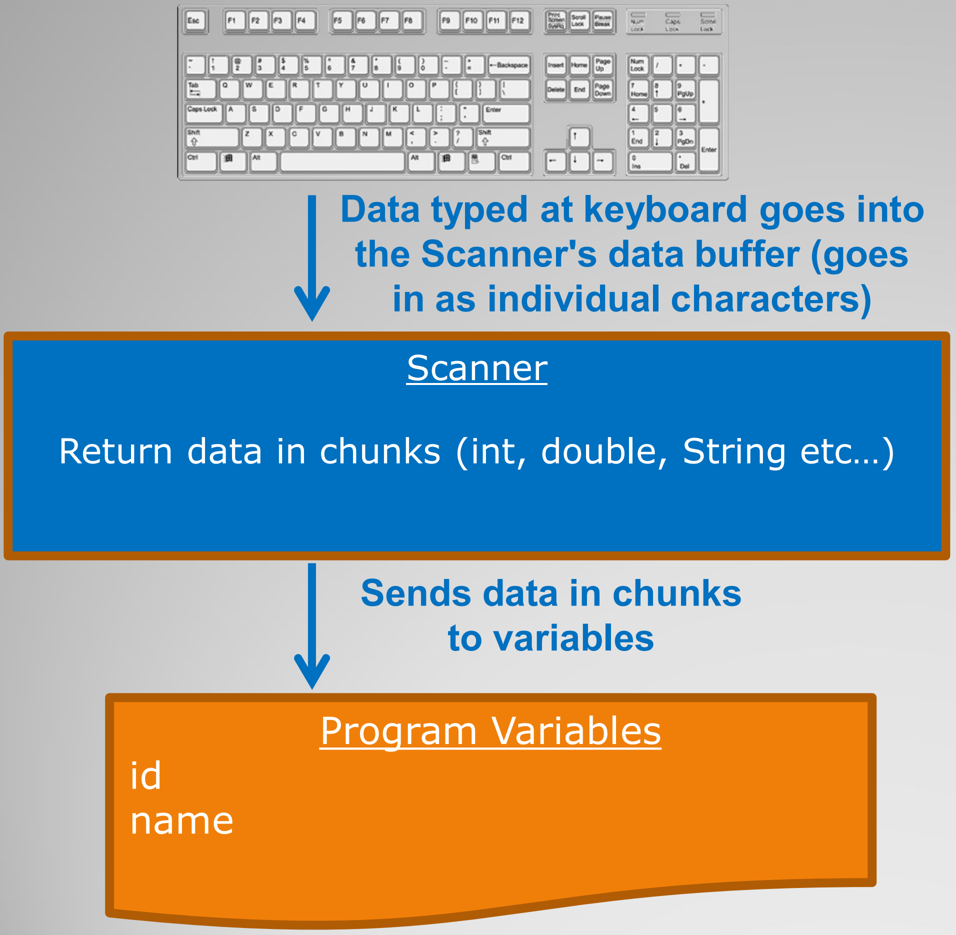
Read from Keyboard

- You need to read data from the keyboard into a program, how do you do it???



Read from Keyboard

- Use a Scanner object to read from the keyboard.



Data typed at keyboard goes into the Scanner's data buffer (goes in as individual characters)
Scanner
Return data in chunks (int, double, String etc...)
Sends data in chunks to variables
Program Variables
id
name

Scanner gets data from keyboard and sends to program variables using next... methods

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

int id = s.nextInt();
String name = s.nextLine();
```

- Code to read in an int:

```
import java.util.Scanner;

public class Welcome1 {

    public static void main( String args[] ) {
            int number1;
            Scanner input = new Scanner(System.in);

            System.out.printf("Enter number: ");
            number1 = input.nextInt();

            System.out.printf("You entered %d", number1);
    }
}
```

Read from standard input

Scanner gets data from keyboard then sends to program variable number1

Read from Keyboard

Read from Keyboard

- Code to read in a string:

```
String s;
Scanner input = new Scanner(System.in);

s = input.nextLine();
```

Read from standard input

Use nextLine() to read a string.

Read from File

- You need to read data from a file into a program, how do you do it???

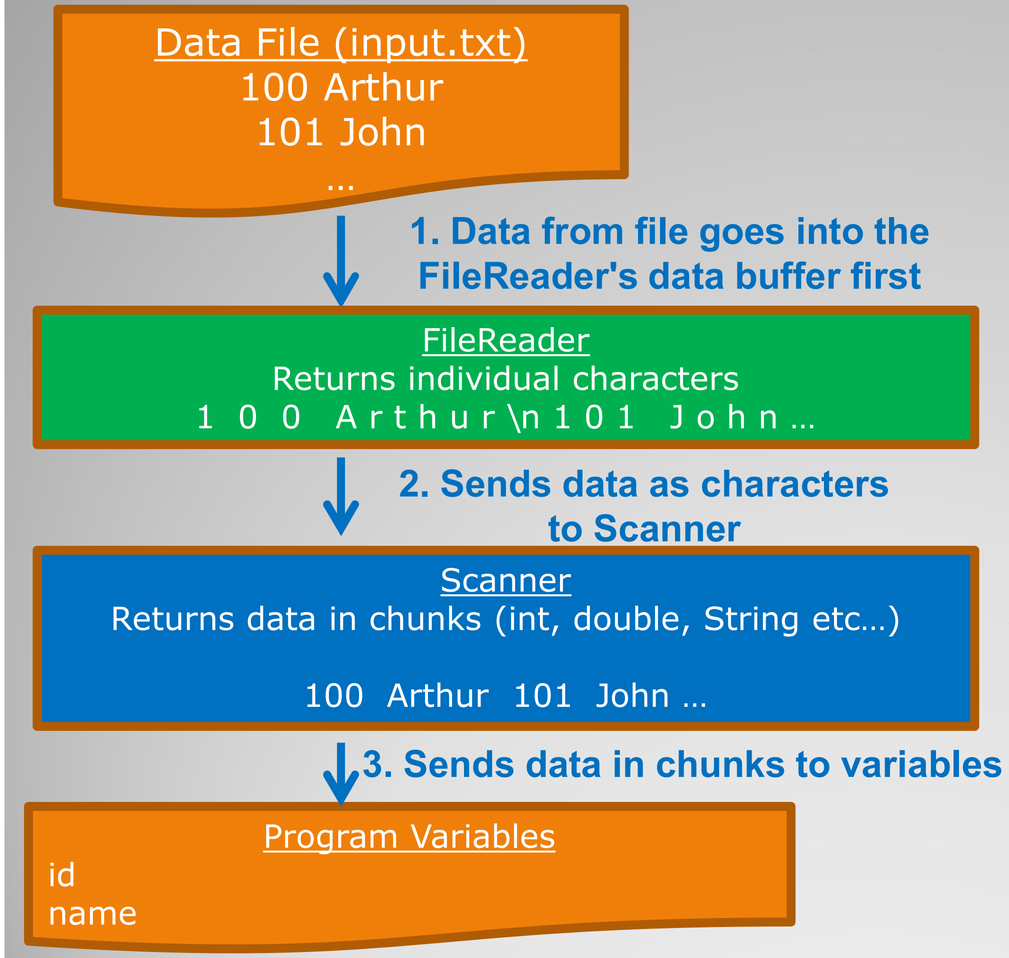
Data File (input.txt)
100 Arthur
101 John
…

???

Program Variables
id
name

Read from File

- Use a FileReader with a Scanner to read from a file.


Data File (input.txt)
100 Arthur
101 John
…
1. Data from file goes into the FileReader's data buffer first
FileReader
Returns individual characters
1 0 0 A r t h u r \n 1 0 1 J o h n …
2. Sends data as characters to Scanner
Scanner
Returns data in chunks (int, double, String etc…)
100 Arthur 101 John …
3. Sends data in chunks to variables
Program Variables
id
name

FileReader gets data from the file and passes it along to the Scanner which then sends to the program variables

```
Scanner s;
int id;
String name;
FileReader fr = new FileReader("input.txt");
s = new Scanner(fr);
id = s.nextInt(); // Read an int
name = s.nextLine(); // Read a string
```

Keep Reading File Data

- To keep reading from a file you can use the Scanner's has methods.

```
Scanner s = new Scanner(new Filereader("input.txt"));
```

Keep reading while there is another string to read

```
while ( s.hasNext() )
{
  // Code to read data goes here…
}
```

- Other has methods: hasNextInt(), hasNextDouble() etc…

Read from File

- Will this properly read from the file input.txt?

```
Scanner s;
s = new Scanner("input.txt");
int id = s.nextInt(); // Read an int
String name = s.nextLine(); // Read a string
```

Read from File

- Will this properly read from the file input.txt? **NO!**

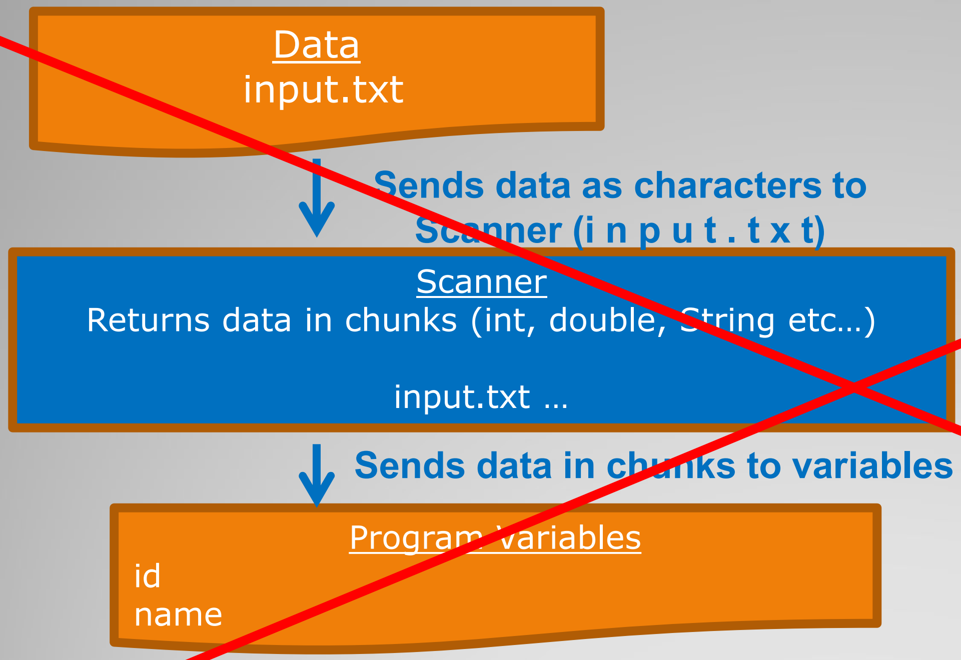
~~Scanner s;~~
~~s = new Scanner("input.txt");~~
~~int id = s.nextInt(); // Read an int~~
~~String name = s.nextLine(); // Read a string~~

- **The string "input.txt" is treated as the actual data and NOT a filename in this case.**
- **The program will crash when nextInt() is called because there are string type data in the input stream.**

Read from File

Read from File

- Incorrect. Uses "input.txt" as the data (not as a filename).

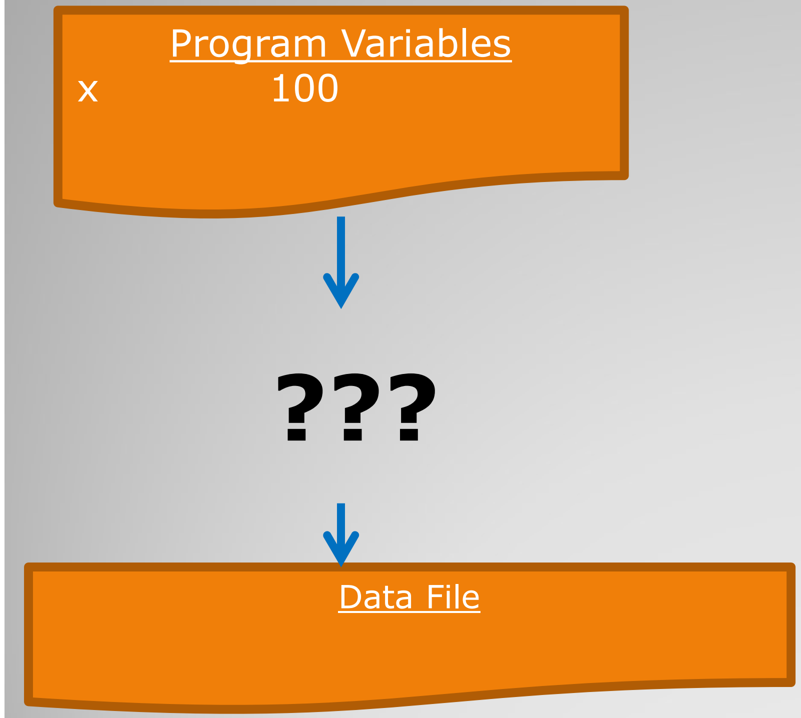

Data
input.txt
Sends data as characters to Scanner (i n p u t . t x t)
Scanner
Returns data in chunks (int, double, String etc…)
input.txt …
Sends data in chunks to variables
Program variables
id
name

The "input.txt" is the actual data in this case

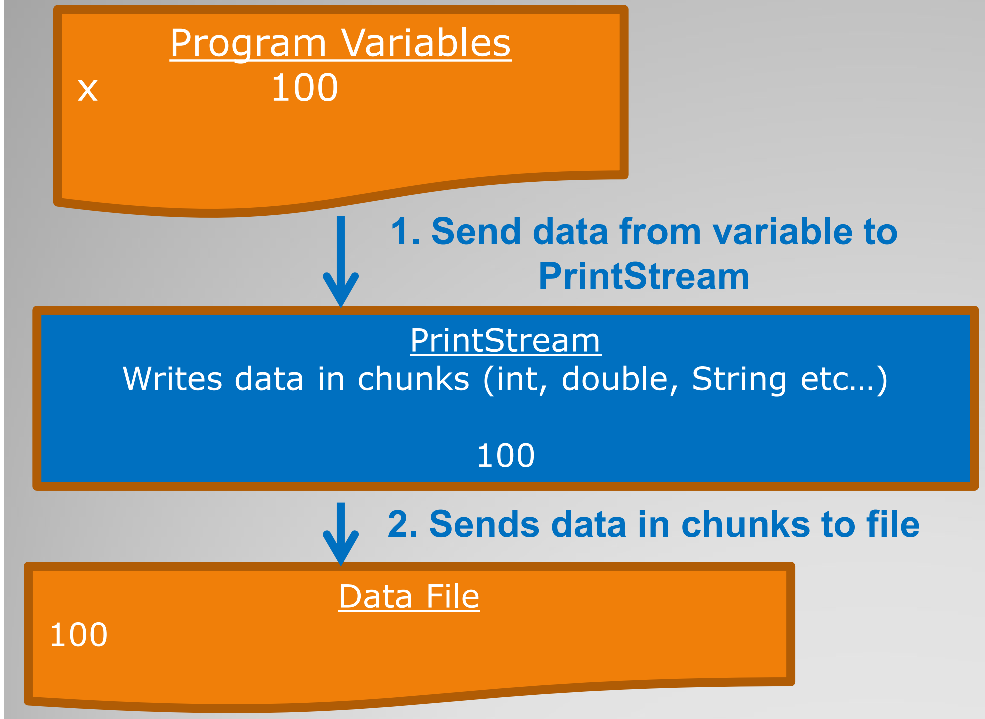
```
Scanner s;
s = new Scanner("input.txt");
int id = s.nextInt(); // Read an int
String name = s.nextLine();
```

Write to a File

- You need to write data to a file from the program, how do you do it???



- Use a FileReader with a Scanner to read from a file.


Program Variables
x 100
1. Send data from variable to PrintStream
PrintStream
Writes data in chunks (int, double, String etc…)
100
2. Sends data in chunks to file
Data File
100

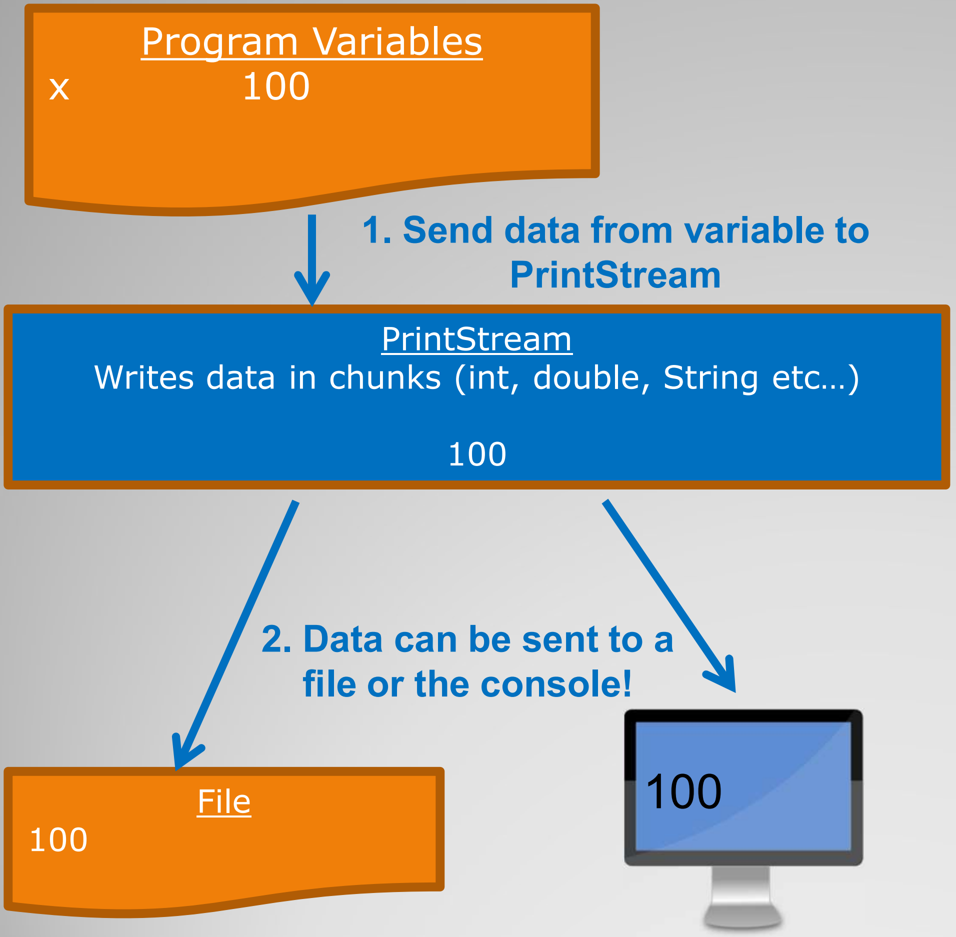
PrintStream gets data from variables and sends to the file

```
int x = 100;
PrintStream ps;
ps = new PrintStream("output.txt");
ps.println(x);
```

Write to a File

PrintStream to Different Places

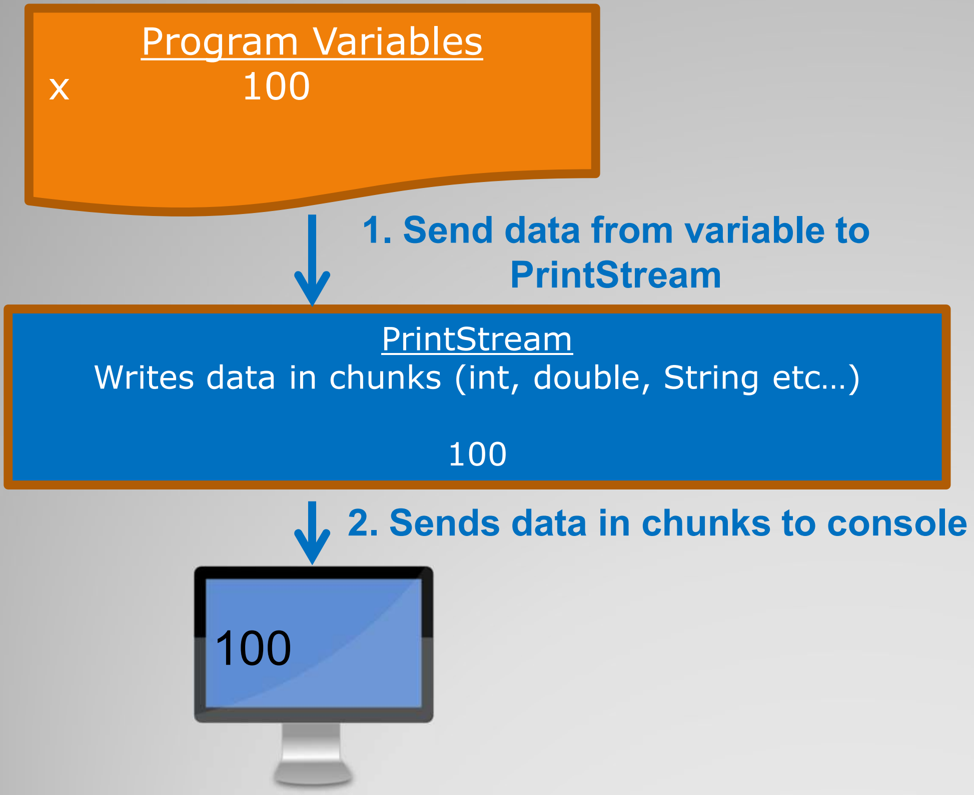
- PrintStream can write to different places (file or console).


Program Variables
x 100
1. Send data from variable to PrintStream
PrintStream
Writes data in chunks (int, double, String etc…)
100
2. Data can be sent to a file or the console!
File
100
100

PrintStream to File

```
int x = 100;
PrintStream ps;
ps = new PrintStream("output.txt");
ps.println(x);
```

- Use a FileReader with a Scanner to read from a file.


Program Variables
x 100
1. Send data from variable to PrintStream
PrintStream
Writes data in chunks (int, double, String etc…)
100
2. Sends data in chunks to console
100

PrintStream gets data from variables and sends to the console

```
int x = 100;
PrintStream ps;
ps = new PrintStream(System.out);
ps.println(x);
```

Write to Console Using PrintStream

- Now we will move on to Java formatted output…

Java Formatted Output

- printf – print formatted

```
System.out.printf("Yanks are number %d", 1);
```

- This statement prints "Yanks are number 1".
- The %d is replaced by 1.
- printf does NOT automatically add a carriage return.

Java Formatted Output

Java Formatted Output

- printf – print formatted (continued)

```
System.out.printf("Yanks are number %d", 1);
```

- The string in double quotes is called a formatted string.
- %d is a "format specifier".

- printf – print formatted (continued)

```
System.out.printf("Yanks are number %d", 1);
```

- Format specifiers are replaced by the arguments that follow the formatted string (the number 1 in this example).
- Again, when this method runs the %d is replaced by 1.
- ***Can we use a variable in place of 1?***

Java Formatted Output

- Yes, you can use a variable argument.

```
int rating=1;
System.out.printf("Yanks are number %d", rating);
```

- Output is the same as before.

- Can we use more than one format specifier?

Java Formatted Output

- Yes, you can use as many format specifiers as you like.

```
String team="Yanks";
int rating=1;

System.out.printf("%s are number %d", team, rating);
```

%s is the string format specifier

%d is the number format specifier

Java Formatted Output

- Add the "\n" to the string to insert a carriage return

```
System.out.printf("Yanks are number 1\n");
```

- You can use as many "\n" as you like.
- "\n" is called an escape sequence

- "\t" gives a tab, "\\" gives a backslash
- There are other escape sequences listed on page 45

Java Formatted Output

Format Specifier	Variable Type
%d	int
%f	float, double
%s	String
%b	boolean

Column Widths (string)

You can set a columns width using printf. This code sets the column width for a String format specifier.

```
String  name = “Arthur”;
System.out.printf(“Name: %20s\n", name);
```

This will display the following (notice the padding after is):

```
Name:              Arthur
```

printf - Format Specifiers

printf - Format Specifiers

Format Specifier	Variable Type
%d	int
%f	float, double
%s	String
%b	boolean

Floating Point Formatting
Show a certain number of places after the decimal point:

```
double num = 10.4567;
System.out.printf("num is %.2f\n", num);
```

This will display the following:
num is 10.46
Note: Java rounds off the number automatically.

Format Specifier	Variable Type
%d	int
%f	float, double
%s	String
%b	boolean

<u>Column Widths (floating point)</u>
You can set a columns width for a floating point format specifier using printf. The following code sets the column width to 20 and decimal places to 2.

```
double num = 10.4567;
System.out.printf("num is %20.2f\n", num);
```

This will display the following (notice the padding after is):

```
num is                10.46
```

printf - Format Specifiers

- Other miscellaneous topics…

Other Miscellaneous Topics

- It is sometimes necessary to put code that opens a file in a try/catch block (do not worry if try/catch is unfamiliar to you now).
- For example, with a PrintStream or a FileReader.
- Here is some code…

Opening Files Within a Try/Catch

```
PrintStream ps = null;

try
{
      ps = new PrintStream("ArthurOutput.txt");
}
catch (Exception e)
{
      System.out.println("ERROR. Could not open file!");
}

// Code to use PrintStream here…
```

Initialize PrintStream to null since declared outside try/catch

Open PrintStream within try/catch block

Opening Files Within a Try/Catch

Opening Files Within a Try/Catch

```
FileReader fr = null;
Scanner s = null;

try
{
	fr = new FileReader("input.txt");
	s = new Scanner(fr);
}
catch (Exception e)
{
	System.out.println("ERROR. Could not open file!");
}

// Code to use Scanner here…
```

Initialize FileReader and Scanner to null since declared outside try/catch

Open FileReader within a try/catch block

print vs println vs printf with newlines

print does NOT go to next line:
```
System.out.print("Yanks");
System.out.print("Great");
```
Output
```
YanksGreat
```

println goes to the next line:
```
System.out.println("Yanks");
System.out.print("Great");
```
Output
```
Yanks
Great
```

printf requires \n for newline (missing \n):
```
System.out.printf("Yanks");
System.out.prinf("Great");
```
Output
```
YanksGreat
```

printf requires \n for newline (has \n):
```
System.out.printf("Yanks\n");
System.out.prinf("Great");
```
Output
```
Yanks
Great
```

print vs println vs printf with newlines

- End of Slides

End of Slides