

NAME _____

DATE _____

PERIOD _____

UNIT 1 · TOPIC 1.7 · WORKSHEET

Check Out the Library

Read the docs, fill the import, sort the facts from the actions, and catch the claim that's bluffing.

About 30 minutes

9 questions

Cumulative through 1.7

TOPIC

1.7

**DIRECTIONS**

Work in pencil. For code, write exactly what prints, one line per `println`. Use the documentation on this sheet, not guesses; calls like `Math.sqrt(area)` follow it, and their full rules come later. Some puzzles include pieces you won't need.

**THINK FIRST**

Your programs use `String` and `Math` with no import line, but `Scanner` needed `import java.util.Scanner`; back in Topic 1.4. Why the difference? Guess in the margin; questions 1 and 8 settle it.

**KEY IDEA****From the Math documentation (simplified)**

Questions 2, 5, and 7 use it. Read past the word `static` for now; a later topic explains it. The word just before each method's name is the type of value it gives back.

```
Package java.lang
Class Math
```

Method Summary

<code>static int</code>	<code>abs(int a)</code>	absolute value of a
<code>static double</code>	<code>pow(double a, double b)</code>	a to the power b
<code>static double</code>	<code>sqrt(double a)</code>	square root of a

PART A · 3 QUESTIONS

A Warm up your brain

1 PENCIL CHECK

This program sells tickets at a school play. Every capitalized class name in it is listed below. For each one, write who wrote it (*library* or *the programmer*), its package (or *none*), and whether the file needs an import line for it.

TicketBooth.java

```
1 import java.util.Scanner;
2
3 public class TicketBooth {
4     public static void main(String[] args) {
5         Scanner console = new Scanner(System.in);
6         int seats = console.nextInt();
7         String show = "Into the Woods";
8         System.out.println(show);
9         System.out.println(Math.abs(seats - 120));
10        System.out.println(Integer.MAX_VALUE);
11    }
12 }
```

(Read past public and static for now; a later topic explains them.)

Class	Who wrote it?	Package	Import line needed?
TicketBooth			
Scanner			
String			
Math			
Integer			



FAIR QUESTION

Why does Java bother with packages at all?

Java's library has thousands of classes. Packages sort them onto labeled shelves, so you can find the one you need, and two classes on different shelves can even share a short name without a clash. Think of a package name as the shelf label and the class name as the book.

2 YOU'RE THE JVM

You are the Java Virtual Machine. Use the documentation on page 1 and run this code in your head. Write exactly what prints.

```
double big = Math.pow(4.0, 3.0);
double root = Math.sqrt(big);
int gap = Math.abs(3 - 10);
System.out.println(big);
System.out.println(root);
System.out.println(gap);
```

OUTPUT

3 YOU'RE THE COMPILER

You are the compiler. For each snippet, decide whether it compiles. If it does, write what it prints. If it doesn't, say why in a few words.

Snippet	Compiles?	Output, or why not
1 int top = Integer.MAX_VALUE(); System.out.println(top);		
2 double d = Math.pow(2.0, 5.0); System.out.println(d);		
3 double r = math.sqrt(81.0); System.out.println(r);		
4 String pet = "Biscuit"; System.out.println(pet.length());		
5 int pet = "Biscuit"; System.out.println(pet);		
6 int low = Integer.MIN_VALUE; System.out.println(low);		

PART B · 3 PUZZLES

B Puzzles

4 CODE TILES

A word-game program reads one word and tells how far its length is from a goal of 8 letters. Someone knocked the inside of main off the fridge; the program already has `import java.util.Scanner;` at the top. Put the tiles back in order so that, when the user types `Lighthouse`, it prints the output below. One tile is a decoy and isn't used.

The `String` documentation lists `int length()`: the number of characters in this string.

<code>System.out.println(Math.abs(8 - letters));</code>	<code>int letters = word.length();</code>
<code>Scanner console = new Scanner(System.in);</code>	<code>int letters = word.length;</code>
<code>String word = console.next();</code>	<code>System.out.println(letters);</code>

Target output

10

2

YOUR PROGRAM

5 SNIPPET POOL

A patio program reads the area of a square patio, finds the length of one side, and then finds the volume of a cube-shaped planter with that same side length (side × side × side). Fill each numbered hole with a snippet from the pool so that, when the user types 49.0, it prints 7.0 and then 343.0. Use the Math documentation on page 1. Each snippet is used at most once, and some aren't needed.

Patio.java

```
import (1);

public class Patio {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        double area = console.nextDouble();
        double side = Math.(2)(area);
        double planter = Math.pow((3));
        System.out.println(side);
        System.out.println(planter);
    }
}
```

THE POOL

Each snippet can be used only once, and not every snippet is needed.

java.lang.Scanner	squareRoot	side, 3.0
java.util.Scanner	abs	3.0, side
java.util	sqr	



WATCH OUT

cannot find symbol is the compiler saying "I don't know that name." Read the symbol: line under it. symbol: class Scanner usually means a missing import; symbol: method area() means a method name nobody wrote. When the documentation doesn't list a name, the name doesn't exist, however sensible it sounds.

6 FACE-OFF

Tonight: an Attribute and a Behavior from the `Integer` and `String` classes argue about who matters more. Fill in the blanks.

ATTRIBUTE	I'm <code>Integer.MAX_VALUE</code> . I'm a piece of _____, stored in a _____. You just read me by name.
BEHAVIOR	Reading is easy. I'm <code>length()</code> . I'm an _____, defined by a _____, and every <code>String</code> object can do me.
ATTRIBUTE	You need _____ every time someone uses you, even when nothing goes inside them. I never do.
BEHAVIOR	And in the documentation, you're stuck in the _____ Summary while I get the _____ Summary.
ATTRIBUTE	Put parentheses after my name and the compiler goes looking for a method that doesn't exist. It says _____.
BEHAVIOR	Fine, you're reliable. But ask <code>"ox"</code> and <code>"elephant"</code> for <code>length()</code> and you get two different answers, because each object holds its own _____.

PART C · 3 QUESTIONS

C Catch the mistake

7 ONE-TOKEN FIX

This code should print `12.0`, but it doesn't compile. Use the `Math` documentation on page 1 to change exactly one token so it prints `12.0`.

```
1 double area = 144.0;
2 int side = Math.sqrt(area);
3 System.out.println(side);
```

LINE, OLD TOKEN, NEW TOKEN, AND WHY

8 TWO TRUTHS AND A LIE

Two of these claims about the code are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
String city = "Denver";
int miles = 412;
double root = Math.sqrt(16.0);
System.out.println(city);
System.out.println(root);
```

- **1** `city` has a reference type, defined by the `String` class, and "Denver" is an object, an instance of that class.
- **2** `miles` has a primitive type, so no object is created on line 2.
- **3** Because `Math` is a library class, this code compiles only if `import java.lang.Math;` is added at the top.

THE LIE, REWRITTEN

9 WHO AM I?

Some library ideas are playing a party game. Each gives a clue about itself. Name the idea.

Clue	I am...
I'm a collection of classes that someone else already wrote and tested.	
I'm a named group of related classes, written in lowercase words joined by dots.	
I tell programmers how to use a class without showing them its code.	
I'm a line at the top of a file that lets the file use a class by its short name.	
I'm one particular thing made from a class, with my own values for its attributes.	