Name: _____ Session: AM / PM
Circle one

Chapter 2 - Day 3

Scope, Style, and Interactive Input

VARIABLE SCOPE

Scope is the part of a program where a variable can be accessed by _____.

A variable declared inside a method is a _____ variable.

A local variable's scope begins at its _____ and ends at the closing brace of its enclosing block.

Reference: Declare a local variable before using it, and assign it a value before reading it.

```
1 public static void main(String[] args)
2 {
3     System.out.println(score); // This line causes an error.
4     int score = 100;
5 }
```

Repair: Which statement must move above the print statement?

THREE KINDS OF COMMENTS

Kind	Syntax	Purpose
Single-line		
Multi-line		
Documentation		

The **Javadoc** tool can turn documentation comments into _____ documentation. Comments explain code to people; they do not run as program statements.

A useful comment explains why code exists, what a section does, or something that is not obvious. Avoid simply restating a Java statement.

READABLE CODE

Use consistent indentation inside braces, meaningful names such as badgeId, spaces around operators, and blank lines between related sections. These habits help people follow the program.

Two programs can compile and produce the same output but differ in _____.

CHAPTER 2 - DAY 3

Scanner & Keyboard Input

IMPORT, CREATE, PROMPT, READ

Scanner lets a program read data while it is running. Put the import before the class declaration. Create the Scanner inside main before using it.

```java
import java.util.Scanner;
public class App
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Enter age: ");
        int age = keyboard.nextInt();
        System.out.print("Enter GPA: ");
        double gpa = keyboard.nextDouble();
        System.out.println("Age: " + age + ", GPA: " + gpa);
    }
}
```

Read the setup: Scanner is the class/type. keyboard is the variable that refers to the Scanner object. the new creates the object; System.in supplies standard input (usually the keyboard).

The Scanner variable name is your choice. Use the same name in every method call.

CHOOSE A METHOD THAT MATCHES THE DATA

Data needed	Scanner call	What it reads
int	keyboard._____	A whole-number token, such as 24.
double	keyboard._____	A numeric token, such as 3.75.
String	`keyboard.nextLine()`	The rest of the current line, including spaces; consumes the line break.

Always show a _____ before reading input.

READING A FULL NAME

```java
System.out.print("Enter your full name: ");
String name = keyboard.nextLine();
System.out.println("Welcome, " + name + "!");
```

If the user enters Bob Fisherman, what value is stored in name?

CHAPTER 2 - DAY 3

The Scanner Trap & Two Fixes

WHY A NAME CAN BE SKIPPED

nextInt() and nextDouble() read the number but leave the rest of the line, including its line break.
nextLine() reads whatever remains on that line.

```
1 int age = keyboard.nextInt();
2 String name = keyboard.nextLine();  // May read an empty String.
```

Suppose the user types **24** and presses **Enter**, with nothing after the number. Here, \n represents the line break.

Moment	Trace the input
Before nextInt()	[2] [4] [\n]
After nextInt()	age = _____ Input remaining: _____
After nextLine()	name = _____ Did it wait for a name? _____

FIX 1: CONSUME THE REST OF THE NUMERIC LINE

```
1 System.out.print("Enter age: ");
2 int age = keyboard.nextInt();
3 keyboard.nextLine();  // Discard the rest of the age line.
4 System.out.print("Enter your full name: ");
5 String name = keyboard.nextLine();
```

The cleanup line stores nothing. Its job is to consume the leftover _____. Use it when switching from a numeric read to reading text on the **next** line. Do not insert cleanup calls after every input; an extra call can consume wanted text.

FIX 2: READ EVERY INPUT AS TEXT, THEN PARSE

Parsing converts text representing a number into a numeric value.
nextLine() always returns a String, even when the user types digits.

Needed type	Conversion example
int	`Integer.parseInt("24")`
double	`Double.parseDouble("3.75")`
long	`Long.parseLong("3000000000")`

```
1 String ageText = keyboard.nextLine();
2 int age = Integer.parseInt(ageText);
```

All-line input needs no cleanup call. Numeric text must fit the chosen type: Integer.parseInt(“3.5”) fails because 3.5 is not a whole number.

Complete the conversion: `double gpa = _____(gpaText);`