Name: _____ Session: AM / PM

Circle one

Chapter 5 - Day 1

Void Methods and Passing Arguments

VOID METHODS

What a method is: A method is a named block of statements that runs when it is called. Methods let you divide a big problem into smaller pieces, and write a task once, then _____ it wherever it is needed. A void method does its task and sends nothing back. A value-returning method sends a value back to the code that called it.

Term	Definition / memory cue
method	
void method	
value-returning method	
method header	
method body	
call statement	

M1 Header, body, and call `M1_MissionBriefing.java`

What changed: main prints a line, calls showBriefing, then prints again. The method is written below main.

```
2  public static void main(String[] args) {
3      System.out.println("Mission Control online.");
4      showBriefing();
5      System.out.println("Back in main.");
6  }
7
8  /**
9   * Displays the mission briefing.
10  */
11 public static void showBriefing() {
12     System.out.println("Today: dock at the station.");
13 }
```

1. The call statement is on line

2. The method header is on line

Calling a method: A call statement never repeats `void` or the modifiers, and it always needs its parentheses, even when they are empty. When the method finishes, the program goes back to the line _____. A call is a statement, so a loop can repeat it: in M2, printStatus runs _____ times. A method can also call another method; this is a hierarchical call.

M3 A method that calls a method

`M3_PreflightCheck.java`

What changed: main calls checkEngines, and checkEngines calls checkFuel. The numbered lines show the order.

```
2  public static void main(String[] args) {
3      System.out.println("1. Begin preflight");
4      checkEngines();
5      System.out.println("5. Preflight done");
6  }
7
8  // checkEngines calls checkFuel: a hierarchical call
9  public static void checkEngines() {
10     System.out.println("2. Engines: checking");
11     checkFuel();
12     System.out.println("4. Engines: ready");
13 }
14
15 public static void checkFuel() {
16     System.out.println("3. Fuel: full");
17 }
```

1. When checkFuel ends, the next line to run is line

2. When checkEngines ends, the next line to run is line

M5 Practice in pairs: trace it

`M5_TraceLetters.java`

What changed: Trace the calls before it runs. Write the six lines it prints, in order.

```
2  public static void main(String[] args) {
3      System.out.println("W");
4      alpha();
5      System.out.println("X");
6      beta();
7  }
8
9  public static void alpha() {
10     System.out.println("M");
11     beta();
12     System.out.println("N");
13 }
14
15 public static void beta() {
16     System.out.println("K");
17 }
```

1. Lines 1 to 3 printed:

2. Lines 4 to 6 printed:

3. beta runs _____ times

ARGUMENTS AND PARAMETERS

Passing an argument: An argument is a value in a call's parentheses. A parameter is the variable in the method header that _____ it. You can pass a literal, a variable, or an expression; the method gets its value.

Term	Definition / memory cue
argument	
parameter	
parameter list	
widening conversion	
narrowing conversion	

P1 Three kinds of argument

`P1_BurnTimer.java`

What changed: displayBurn is called three times: with a literal, a variable, and an expression.

```
2  public static void main(String[] args) {
3      int seconds = 12;
4      displayBurn(5);           // a literal
5      displayBurn(seconds);     // a variable
6      displayBurn(seconds * 2); // an expression
7  }
8
9  public static void displayBurn(int time) {
10     System.out.println("Burn: " + time + " seconds");
11 }
```

1. The parameter is named

2. Line 6 passes the value

Two or more parameters: Separate parameters with commas, and give _____ its own data type: `(int high, int low)`, never `(int high, low)`. Arguments are matched to parameters by position: the first argument goes to the first parameter.

P2 Two parameters

P2_OrbitGap.java

What changed: showDock takes a String and an int. showGap is called twice with the same values in swapped order.

```
2  public static void main(String[] args) {
3      showDock("Orion", 2);
4      showGap(15, 4);
5      showGap(4, 15); // same values, swapped order
6  }
7
8  public static void showDock(String craft, int port) {
9      System.out.println(craft + " at port " + port);
10 }
11
12 public static void showGap(int high, int low) {
13     System.out.println("Gap: " + (high - low));
14 }
```

1. In showDock, port receives

2. Line 5 prints Gap:

Argument types: Java widens a `byte` or `short` argument to `int` automatically, because no data is lost. A `double` argument for an `int` parameter needs a _____, such as `(int) precise`, which drops the decimal part.

P3 Widening and a cast

P3_ThrustTypes.java

What changed: A byte and a short are passed to an int parameter, then a double with a cast.

```
2  public static void main(String[] args) {
3      byte small = 8;
4      short medium = 300;
5      double precise = 4.9;
6      showThrust(small);         // byte widens to int
7      showThrust(medium);        // short widens to int
8      showThrust((int) precise); // needs a cast
9  }
10
11 public static void showThrust(int kN) {
12     System.out.println("Thrust: " + kN + " kN");
13 }
```

1. Line 8 prints Thrust: _____ kN

2. Without (int), line 8 would

Practice in pairs. The header is public static void setAltitude(int feet), and the variables are byte b = 9; short s = 1200; long big = 5000; float f = 2.5f; double d = 30.0;

Call	Compiles?	Why
`setAltitude(b);`		
`setAltitude(s);`		
`setAltitude(big);`		
`setAltitude(f);`		
`setAltitude(d);`		
`setAltitude((int) d);`		

PASS BY VALUE AND STRINGS

Pass by value: Java passes a primitive argument by value: the parameter gets a _____ of the argument's value. Changing the parameter inside the method never changes the caller's variable.

Term	Definition / memory cue
pass by value	
reference variable	
immutable	

V1 Changing a copy

`V1_FuelCopy.java`

What changed: refuel adds 60 to its parameter. main prints fuel before and after the call.

```
2  public static void main(String[] args) {
3      int fuel = 40;
4      System.out.println("Before: " + fuel);
5      refuel(fuel);
6      System.out.println("After: " + fuel);
7  }
8
9  // tank gets a copy of fuel's value
10 public static void refuel(int tank) {
11     tank = tank + 60;
12     System.out.println("Inside refuel: " + tank);
13 }
```

1. Line 11 makes tank

2. Line 6 prints After:

Passing a String: A String variable is a reference variable, so passing it gives the parameter the _____. Strings are immutable: a method can't change their text. `name = "Apollo";` only points the parameter at a different String; the caller's variable still refers to the original.

V2 Passing a String

`V2_ShipName.java`

What changed: showLength reads the String it was passed. rename points its parameter at a new String.

```
2  public static void main(String[] args) {
3      String ship = "Artemis";
4      showLength(ship);
5      rename(ship);
6      System.out.println("Still: " + ship);
7  }
8
9  // name refers to the same String object as ship
10 public static void showLength(String name) {
11     System.out.println(name + ": " + name.length());
12 }
13
14 // Points name at a new String; ship is not changed
15 public static void rename(String name) {
16     name = "Apollo";
17     System.out.println("Inside rename: " + name);
18 }
```

1. Line 4's call prints Artemis:

2. Line 6 prints Still:

V3 Practice in pairs: trace it

`V3_CrewUpdate.java`

What changed: update changes an int parameter and a String parameter. Predict both lines before it runs.

```
2  public static void main(String[] args) {
3      int crew = 4;
4      String pilot = "Ride";
5      update(crew, pilot);
6      System.out.println(crew + " " + pilot);
7  }
8
9  public static void update(int c, String p) {
10     c = c + 2;
11     p = "Jemison";
12     System.out.println(c + " " + p);
13 }
```

1. Line 12 prints

2. Line 6 prints