**Name:** _____ Session: AM / PM

Circle one

# Chapter 5 - Day 2

## Local Variables and Return Values

### LOCAL VARIABLES

**Local variables:** A local variable is declared inside a method. Its scope is that method: no other method can use it. Its lifetime is one call: it is created when the method runs and _____ when the method ends. A local variable gets no default value, so it must be given a value before it is used.

| Term | Definition / memory cue |
|---|---|
| local variable | |
| scope | |
| lifetime | |

### L1 The same name in two methods

`L1_TwoStations.java`

**What changed:** stationAlpha and stationBeta each declare a local variable named crew.

```
2  public static void main(String[] args) {
3      stationAlpha();
4      stationBeta();
5  }
6
7  public static void stationAlpha() {
8      int crew = 6; // local to stationAlpha
9      System.out.println("Alpha crew: " + crew);
10 }
11
12 public static void stationBeta() {
13     int crew = 3; // a different variable
14     System.out.println("Beta crew: " + crew);
15 }
```

1. Line 9 prints Alpha crew:

2. The two crew variables share only their

### L4 A parameter outside its method

L4_ScopeError.java

**What changed:** main tries to print thrust, a parameter of addThrust. Predict what happens before it compiles.

```
2  public static void main(String[] args) {
3      addThrust(40);
4      System.out.println(thrust);
5  }
6
7  public static void addThrust(int thrust) {
8      thrust = thrust + 25;
9  }
```

1. Line 4 is a _____ error
2. thrust can be used only inside

### L3 Practice in pairs: called three times

L3_LogEntry.java

**What changed:** logEntry is called three times from a loop. Predict the output before it runs.

```
2  public static void main(String[] args) {
3      for (int i = 1; i <= 3; i++) {
4          logEntry();
5      }
6  }
7
8  public static void logEntry() {
9      int count = 0;
10     count++;
11     System.out.println("Entry " + count);
12 }
```

1. Each line printed is Entry
2. count starts at _____ on every call
3. Why: count's _____ is one call

## RETURNING A VALUE

**A value-returning method:** Its header names the type of value it sends back, such as `int`, where a void method has `void`. A `return` statement ends the method and sends a value back to the _____. The value must match the return type: Java never narrows it automatically.

| Term | Definition / memory cue |
|---|---|
| return type | |
| return statement | |

## R1 Sending the value back

R1_ThrustReturn.java

**What changed:** The fix for L4: addThrust returns the new thrust, and main stores it.

```
2  public static void main(String[] args) {
3      int newThrust = addThrust(40);
4      System.out.println(newThrust);
5  }
6
7  // The fix for L4: send the new value back
8  public static int addThrust(int thrust) {
9      return thrust + 25;
10 }
```

1. addThrust's return type is

2. Line 4 prints

**Using a return value:** A call to a value-returning method can go anywhere a value of its type can: in an assignment, inside `println`, or in a larger expression. Storing it in a variable first is _____.

## R2 Using a return value

R2_StageSum.java

**What changed:** sum is used three ways: stored in total, used in an expression, and divided by 2.0.

```
2  public static void main(String[] args) {
3      int x = 7, y = 5;
4      int total = sum(x, y); // store it
5      System.out.println("Total: " + total);
6      System.out.println(sum(x, y) + 1);   // expression
7      System.out.println(sum(x, y) / 2.0); // average
8  }
9
10 /**
11  * Adds two stage burn times.
12  * @param a The first burn time.
13  * @param b The second burn time.
14  * @return The sum of a and b.
15  */
16 public static int sum(int a, int b) {
17     int result = a + b;
18     return result;
19 }
```

1. Line 6 prints

2. Line 7 prints

### R3 Practice in pairs: what's wrong? — R3_WhatsWrong.java

**What changed:** This file has three errors. Find each one, and how to fix it, before it compiles.

```
2  public static void main(String[] args) {
3      System.out.println(sum(6, 2));
4      int average = getAverage(7, 4);
5      System.out.println(halfBurn(9));
6  }
7
8  public static int sum(int a, int b) {
9      int result = a + b;
10 }
11
12 public static double getAverage(int a, int b) {
13     return (a + b) / 2.0;
14 }
15
16 public static int halfBurn(int seconds) {
17     return seconds / 2.0;
18 }
```

1. Line 4:

2. Lines 8-10:

3. Line 17:

## BOOLEAN AND STRING RETURNS

**Returning a boolean:** A method can return `true` or `false`, and its call can be used directly as an `if` condition. `if (isSafeAltitude(altitude))` runs the `if` part when the method returns _____.

### B1 A boolean return as a condition — B1_SafeAltitude.java

**What changed:** isSafeAltitude returns whether the altitude is from 500 to 3000 feet.

```
2  public static void main(String[] args) {
3      int altitude = 3500;
4      if (isSafeAltitude(altitude)) {
5          System.out.println("Safe to deploy.");
6      } else {
7          System.out.println("Hold deployment.");
8      }
9      System.out.println(isSafeAltitude(1200));
10 }
11
12 /**
13  * Checks if the parachute can open.
14  * @param feet The altitude in feet.
15  * @return true if feet is from 500 to 3000.
16  */
17 public static boolean isSafeAltitude(int feet) {
18     return feet >= 500 && feet <= 3000;
19 }
```

1. Line 4's call returns

2. Line 9 prints

**Returning a String:** A `String` method sends back a String, often one it builds: `return first + " " + last;`. `substring(start, end)` returns the characters from `start` up to, but not including, `end`, so `substring(0, 1)` is the _____.

| Term | Definition / memory cue |
|---|---|
| substring(start, end) | |
| @param tag | |
| @return tag | |

## S1 Returning a String

`S1_CrewTag.java`

**What changed:** fullName builds a full name, and initials uses substring on both names.

```
2  public static void main(String[] args) {
3      String name = fullName("Sally", "Ride");
4      System.out.println(name);
5      System.out.println(initials("Mae", "Jemison"));
6  }
7
8  /**
9   * Builds a full name.
10  * @param first The first name.
11  * @param last The last name.
12  * @return The full name.
13  */
14 public static String fullName(String first, String last) {
15     return first + " " + last;
16 }
17
18 public static String initials(String first, String last) {
19     return first.substring(0, 1) + last.substring(0, 1);
20 }
```

1. Line 4 prints

2. Line 5 prints

**Documentation tags:** In a `/** */` comment, write one `@param` line for each parameter and one `@return` line if the method returns a value. A void method has _____ `@return` tag.

Practice in pairs. Write the header for each method. Every method is public static.

| What the method does | Header |
| --- | --- |
| Returns whether a fuel level (a double) is at least 25 percent | |
| Returns a badge made from a first and last name | |
| Returns the total mass of two modules (doubles) | |
| Prints a launch banner and sends nothing back | |