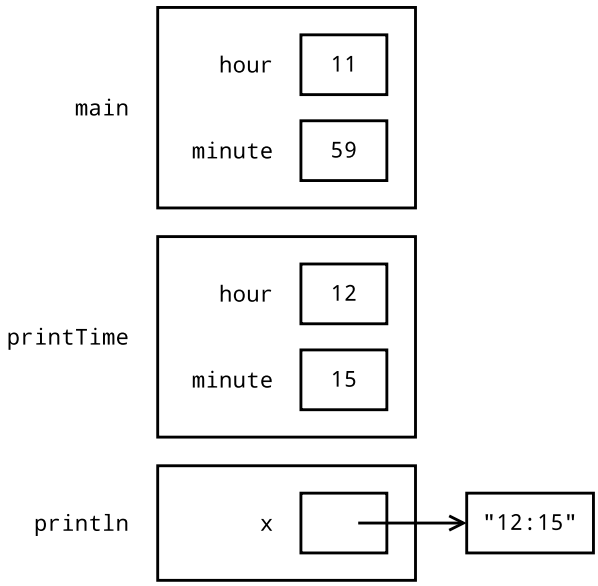
Model 1 Stack Diagrams

The following program displays the time `12:15` on the screen:

```
public static void printTime(int hour, int minute) {
    System.out.println(hour + ":" + minute);
}

public static void main(String[] args) {
    int hour = 11;
    int minute = 59;
    printTime(12, 15);
}
```

We can draw a ***stack diagram*** to visualize how the program works. Each method has its own area of memory to store parameters and other variables. For convenience, we stack method calls underneath each other (upside down).



Questions (15 min) Start time:

1. Based on the diagram, how many methods does the program use? Three

2. Based on the diagram, how many variables does the program have? Five

3. How are stack diagrams different from the memory diagrams we've seen previously?

There is a box for each method call, with the name of the method on the left.

Notice that reference types (like strings) are not stored "on the stack" — the variables are part of the method, but the actual data is stored outside of the method.

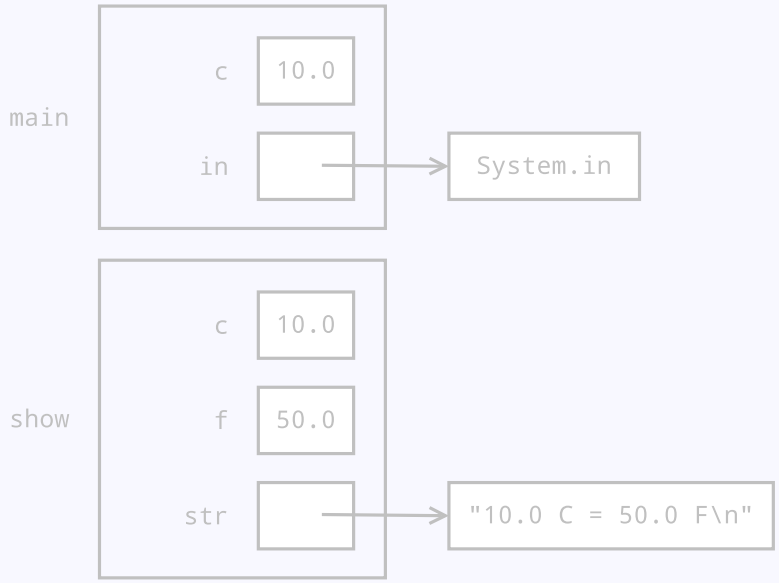
4. How is it possible that two variables with the same name can have different values?

Each method declares its own variables. Because they are stored in different memory locations, the values are independent from other methods.

5. Draw a stack diagram to show the state of memory just before `println` is called. Assume the user inputs the value 10. (You should be able to do this kind of math without a calculator.)

```
public static void show(double c) {
    double f;
    String str;
    f = c * 1.8 + 32;
    str = String.format("%.1f C = %.1f F\n", c, f);
    System.out.println(str);
}

public static void main(String[] args) {
    double c;
    Scanner in = new Scanner(System.in);
    System.out.print("Enter temperature in Celsius: ");
    c = in.nextDouble();
    show(c);
}
```



6. What is the difference between the `String.format` method (used in the previous question) and `System.out.printf`?

They both substitute values based on format specifiers. `String.format` returns a new string, whereas `System.out.printf` displays it on the screen. (In fact, `System.out.printf` invokes the `String.format` method.)