Activity 4: Methods

Java programs are organized into *classes*, each of which has one or more *methods*, each of which has one or more *statements*. Writing methods allows you to break down a complex program into smaller blocks of reusable code.

Model 1   Invoke and Return

Each statement in this program *invokes* (or calls) a method. At the end of a method, Java *returns* to where it was invoked. The list of events on the right illustrates how the program runs.

```
1  public class Model {
2
3      public static void main(String[] args) {
4          System.out.println("First line.");
5          threeLine();
6          System.out.println("Second line.");
7      }
8
9      public static void newLine() {
10         System.out.println();
11     }
12
13     public static void threeLine() {
14         newLine();
15         newLine();
16         newLine();
17     }
18
19 }
```

```
INVOKE println
RETURN to line 5
INVOKE threeLine
    INVOKE newLine
        INVOKE println
        RETURN to line 11
    RETURN to line 15
    INVOKE newLine
        INVOKE println
        RETURN to line 11
    RETURN to line 16
    INVOKE newLine
        INVOKE println
        RETURN to line 11
    RETURN to line 17
RETURN to line 6
INVOKE println
RETURN to line 7
```

Questions (15 min)   Start time: _____

1. How many lines of source code invoke the `println` method?

2. How many times is `println` invoked when the program runs?

3. For each `INVOKE` on the right, draw an arrow to the corresponding line of code. (Plan ahead so that crossing lines will still be legible.)

4. What is the output of the program? Please write \n to show each newline character.

5. When Java sees a name like `x`, `count`, or `newLine`, how can it tell whether it's a variable or a method? (Hint: syntax)

6. What is the difference between a method and a variable? What do they have in common?

7. In your own words, describe what methods are for. Why not just write everything in `main`?

Model 2 Math Methods

Consider the following methods defined in the `Math` class:

```java
public static int abs(int a)
public static double log(double a)
public static double pow(double a, double b)
public static double random()
public static int subtractExact(int x, int y)
```

Note this list isn't exhaustive; `Math` has over 70 methods in total. To invoke methods from another class (like `Math`), you must first specify the class name:

```java
value = abs(-5);        // Error: cannot find symbol
value = Math.abs(-5);   // correct
```

The period in this example is called the *dot operator*. When reading the above code out loud, you would say "math dot abs".

Here is the Java documentation for the methods listed above:

Modifier and Type	Method and Description
static int	**abs**(int a) Returns the absolute value of an int value.
static double	**log**(double a) Returns the natural logarithm (base *e*) of a double value.
static double	**pow**(double a, double b) Returns the value of the first argument raised to the power of the second argument.
static double	**random**() Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.
static int	**subtractExact**(int x, int y) Returns the difference of the arguments, throwing an exception if the result overflows an int.

Questions (15 min)   **Start time:** ⎯⎯⎯⎯⎯⎯

8. What type of value does `Math.random()` return? Give an example of what it would look like.

9. When *invoking* a method, what do you need to specify before the method name and after the method name?

10. For each method, write a Java statement that invokes it and assigns the result to a variable.

11.   When *defining* a method, what do you need to specify before the method name and after the method name?

12.   Define a method named `average` that requires two integers `x` and `y` and returns a `double`. (Don't write any semicolons or braces.)

> *What you wrote for the previous question is called the method's* **signature**. *The variables declared inside the parentheses are called* **parameters**. *When invoking the method, the values you provide are called* **arguments**. *Since arguments will be assigned to parameters, their types must be compatible.*

13.   How many parameters and arguments does each method have? In the table below, what is the relationship between the last two columns?

Method	# Params	# Args
abs		
log		
pow		
random		
subtractExact		
println		

14.   Consider the statement `System.out.println("Price: " + price);` where the value of `price` is 9.99. What is the argument that `println` receives?

15.   Consider the statement `System.out.printf("Price: %f", price);` where the value of `price` is 9.99. Why does `println` use *plus* and `printf` use *comma* to specify the arguments?

Model 3   Stack Diagrams

Each method has its own area of memory to store parameters and other variables. When a method is invoked, Java allocates this memory on the *call stack*. For convenience, we draw "stack" diagrams upside down.
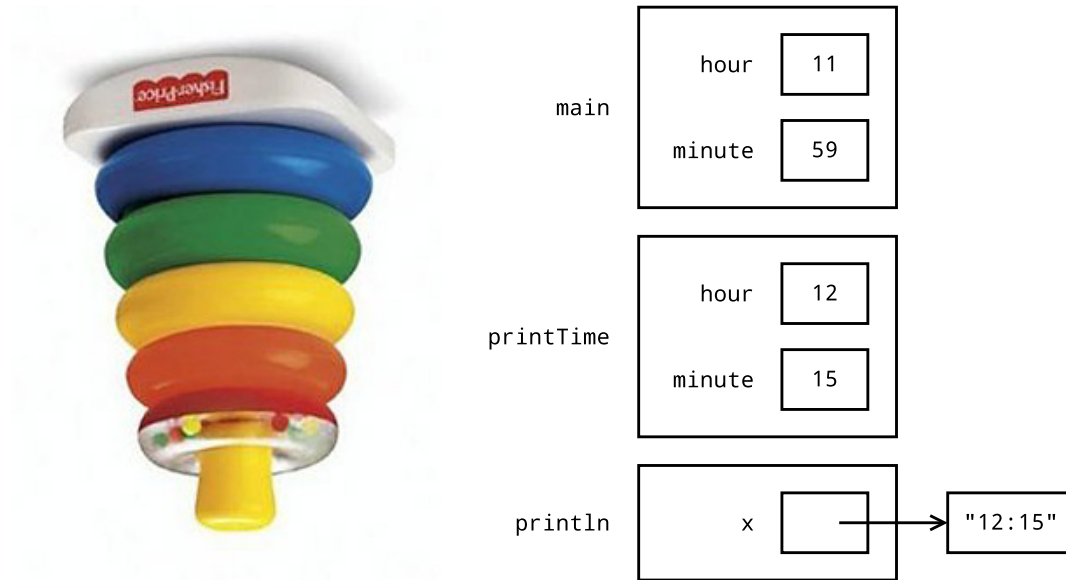

```java
public static void printTime(int hour, int minute) {
    System.out.println(hour + ":" + minute);
}

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

Questions (15 min)   Start time: _____

16. Based on the diagram, how many methods does the program call?

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

18. How do stack diagrams extend the state diagrams we've seen previously?

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

20. 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.)

```java
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);
}
```

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