Practical Coding in Java

Learn to write and validate your own code

Darren Kessner, PhD

(revised September 1, 2025)

Contents

Introduction

This book is a practical guide for learning to write code, using the Java language. It can be used in an introductory Java class (e.g. AP Computer Science), or as a guidebook for self-study.

I emphasize writing unit tests to validate your own code. Writing tests for your own code will give you confidence that your code does what you expect it to. With the development of generative AI, it is easy to generate code to solve any problem in any programming language, and the code may *seem* to work. However, it is now even more important for software developers to ensure that code behaves as expected.

Rather than *reading* this book, I recommend that you focus on *writing code*, i.e. doing the exercises for each chapter. The only way to learn to write code is to actually write code. Each chapter has working code examples that illustrate new syntax or concepts, together with the output from running the code. You will learn much more by struggling with and completing the exercises than by reading the examples.

About the book

This book is based on notes, demo code, and coding exercises I have written and used over the past 10 years of teaching AP Computer Science at Marlborough School in Los Angeles.

Various versions of this content have been published previously on my class webpages with open source licensing,

About the author

I am the Program Head of Computer Science and Software Innovation at Marlborough School in Los Angeles, where I have taught Math and Computer Science for 11 years.

I am also a software developer with over 25 years of experience writing software in a wide variety of fields, including computer security, computer graphics, and scientific applications. My published academic papers include contributions to the areas of bioinformatics, proteomics, and population genetics.

I am a strong proponent of free and open source software, open public data, and open educational resources. I am also an advocate for increasing the diversity of voices in the STEM fields in general, and in software development in particular.

In the classroom I use free and open source software, open public data, and open educational resources.

Darren Kessner, PhD
https://dkessner.github.io

Resources

There are lots of free resources available online for learning Java.

David J Eck, Introduction to Programming Using Java, Seventh Edition
http://math.hws.edu/javanotes/

Wikibooks Java Programming
https://en.wikibooks.org/wiki/Java_Programming

CodingBat code practice
https://codingbat.com/java

0. Hello, world!

This zeroth chapter is a quick overview of Java syntax, including declaring variables, loops (`for`), and conditions (`if`).

One of your main goals this chapter is to write a FizzBuzz program (see Coding Exercises).

Hello

```
//
// Hello.java
//

public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello, world!");
    }
}
```

Output:

```
Hello, world!
```

Basics

```java
//
// Basics.java
//


public class Basics
{
    public static void main(String[] args)
    {
        System.out.println("Hello, world!");

        // basic types: int, float, double, boolean

        int n = 5;
        System.out.println(n);

        n = 7;
        System.out.println(n);

        float x = 1.23f; // floating point
        System.out.println(x);

        double y = 1.23;
        System.out.println(y);

        double my_number = 5.67;  // snake case
        System.out.println(my_number);

        double myNumber = 5.67;  // camel case
        System.out.println(myNumber);

        // this is a comment

        /*
        this is a multiline comment
        this is a multiline comment
        */

        boolean isHappy = true;
```

```
        System.out.println(isHappy);

        // isHappy = 5; // error

        // reference types:  String, ...

        String hello = "Hello, world!";
        System.out.println(hello);

        hello = "blah";
        System.out.println(hello);
    }
}
```

Output:

```
Hello, world!
5
7
1.23
1.23
5.67
5.67
true
Hello, world!
blah
```

Loops

```java
//
// Loops.java
//


public class Loops
{
    public static void main(String[] args)
    {
        System.out.println("Loops");

        // initialization: int i=0
        // condition: i<10
        // update: i++

        // || means or
        // && means and

        for (int i=0; i<10; i++)
        {
            if (i%2 == 0)
            {
                System.out.println("Even");
            }
            else if (i == 7 || i == 3)
            {
                System.out.println("Lucky");
            }
            else
            {
                System.out.println(i);
            }
        }
    }
}
```

Output:

```
Loops
```

```
Even
1
Even
Lucky
Even
5
Even
Lucky
Even
9
```

Sequences

```
//
// Sequences.java
//


public class Sequences
{
    public static void main(String[] args)
    {
        System.out.println("Sequences");
        System.out.println();

        // print multiples of 7

        for (int i=0; i<30; i++)
        {
            if (i%7 == 0)
                System.out.println(i);
        }

        System.out.println();

        // Note:
        //   =  is variable assignment
        //   == is equality comparison

        // Note:
        // i += 7 means:  i = i+7
        // i++ means:  i+=1 (or i=i+1)

        // print multiples of 7 again

        for (int i=0; i<30; i+=7)
            System.out.println(i);

        System.out.println();

        // arithmetic sequence:
        // 3, 10, 17, 24, ...
```

```
        // print sequence using explicit formula

        for (int i=0; i<5; i++)
            System.out.println(i*7 + 3);

        System.out.println();

        // print sequence using recursive formula

        int value = 3;

        for (int i=0; i<5; i++)
        {
            System.out.println(value);
            value += 7;
        }

        System.out.println();
    }
}
```

Output:

```
Sequences

0
7
14
21
28

0
7
14
21
28

3
10
```

```
17
24
31

3
10
17
24
31
```

Coding Exercises: Hello

1. Multiples of 3

Write a program that prints the first 10 multiples of 3. You should write a class `MultiplesOf3` in a file `MultiplesOf3.java`

2. FizzBuzz

Write a FizzBuzz program. Your class should be named `FizzBuzz` and your source file should be named `FizzBuzz.java`.

Your program should iterate through the first 30 positive integers, printing each one. However, if the integer n is a multiple of 3, print `Fizz` instead of the number. And if n is a multiple of 5, print `Buzz` instead. And if n is a multiple of both 3 and 5, print `FizzBuzz` instead.

Sample output:

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
.
.
.
```

3. Geometric sequence

Write a program `Geometric.java` that prints out the first terms of a geometric sequence, i.e. a sequence with a common ratio, for example: 3,

6, 12, 24, 48, ...

4. Cubes

Write a program `Cubes.java` that prints out the cubes of the counting numbers: 0, 1, 8, 27, 64, 125, ...

5. Fibonacci sequence

Write a program `Fibonacci.java` that prints out the first 30 terms of the Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Hint: I find it easiest to think about this problem using 3 variables: a and b slide up the sequence, and we use a temporary variable c to help do this.

Challenge: After you've done this exercise, try doing it using only 2 variables.

Challenge: Try printing out the ratios of successive terms of the Fibonacci sequence. The sequence of ratios approaches a limit - do you recognize what this limit is?

1. Functions and Testing

In this chapter your goal is to become familiar with functions, and writing unit tests to test your functions. This is the most important chapter of the book.

Unit tests are low-level tests of a single function (the target function). In the unit test function, you are given sample input and the expected output (return value). The unit test function runs the target function with the given input and checks that the return value matches what was expected.

Unit test functions are important for verifying the behavior of your code. As you add more code to your project, it is easy to introduce new bugs. Running the unit tests after any changes will give you confidence that your functions still behave as you expect.

Unit tests are also important for ongoing maintenance of your code. For example, if you find a new bug, you can:

- write a new unit test that fails due to the bug
- fix the bug
- verify that all unit tests pass

In the software development world, this procedure of writing tests first is called *test driven development (TDD)*.

Functions

```
//
// Functions.java
//


public class Functions
{
    public static void hello()
    {
        System.out.println("Hello, world!");
    }

    public static double sum(double a, double b)
    {
        return a+b;
    }

    public static String twice(String s)
    {
        return s+s;
    }

    public static boolean isOdd(int n)
    {
        if (n%2 == 1)
        {
            // another line
            return true;
        }
        else
            return false;
    }

    public static void main(String[] args)
    {
        hello();

        double a = 5.0;
        double b = 7.0;
```

```
        System.out.println("a+b: " + sum(a, b));

        double c = sum(6.23, 7.3412345);
        System.out.println("a+b: " + c);

        String result = twice("Hello, APCS! ");
        System.out.println(result);

        System.out.println("isOdd(5): " + isOdd(5));
        System.out.println("isOdd(7): " + isOdd(7));
        System.out.println("isOdd(8): " + isOdd(8));

    }
}
```

Output:

```
Hello, world!
a+b: 12.0
a+b: 13.5712345
Hello, APCS! Hello, APCS!
isOdd(5): true
isOdd(7): true
isOdd(8): false
```

MonkeyTrouble

```
//
// MonkeyTrouble.java
//


public class MonkeyTrouble
{
    public static boolean monkeyTrouble(boolean aSmile,
                                        boolean bSmile)
    {
        return aSmile == bSmile;
    }

    public static void testMonkeyTrouble(boolean aSmile,
                                         boolean bSmile,
                                         boolean expected)
    {
        boolean result = monkeyTrouble(aSmile, bSmile);

        System.out.print("aSmile: " + aSmile +
                " bSmile: " + bSmile +
                " expected: " + expected +
                " result: " + result + " ");

        if (result == expected)
            System.out.println("YAY!");
        else
            System.out.println("Boohoo!");
    }

    public static void main(String[] args)
    {
        System.out.println("Hello, world!");

        testMonkeyTrouble(true, true, true);
        testMonkeyTrouble(false, false, true);
        testMonkeyTrouble(true, false, false);
        testMonkeyTrouble(false, true, false);
    }
```

```
}
```

Output:

```
Hello, world!
aSmile: true bSmile: true expected: true result: true YAY!
aSmile: false bSmile: false expected: true result: true YAY!
aSmile: true bSmile: false expected: false result: false YAY!
aSmile: false bSmile: true expected: false result: false YAY!
```

Coding Exercises: Functions and Testing

1. Vampire

A person is a vampire if she is asleep during waking hours (6:00 to 22:00), or awake during sleeping hours (before 6:00 or after 22:00). Write a class with a static function `boolean isVampire(float hour, boolean awake)` where `hour` is the time represented as a `float` (e.g. 6.5 means 6:30), and `awake` represents whether the person is awake, returning `true` if that person is a vampire. Most importantly, write a unit test function and several tests.

2. Good Deal

A store has marked down the prices of many items, but you only want to buy something if the discount is more than 25% (or in other words, the sale price is $< 75\%$ of the original price). Write a function `boolean goodDeal(double originalPrice, double salePrice)` that returns true if you're getting a good deal on the item. Most importantly, write a unit test function and several tests.

3. (Challenge) Prime Numbers

Write a program to print the prime numbers.

To do this, first write a function `isPrime()`:

```
static boolean isPrime(int n)
{
    // return true <-> n is prime
}
```

Write a unit test function and several unit tests for `isPrime()`.

Then in your `main()` function, loop through the first 100 integers and print only the ones for which `isPrime()` returns true.

2. Strings and Math

This chapter demonstrates the use of strings and math functions in Python.

HelloStrings

```
//
// HelloStrings.java
//


public class HelloStrings
{
    public static void main(String[] args)
    {
        String hello = "HelloWorld";
        System.out.println("hello: " + hello);

        // A String variable is a reference to a String object,
        // which has methods (functions) like `length()`

        System.out.println("hello.length(): " + hello.length());

        // String indexing is 0-based

        System.out.println("hello.charAt(0): " + hello.charAt(0));
        System.out.println("hello.charAt(1): " + hello.charAt(1));
        System.out.println("hello.charAt(2): " + hello.charAt(2));

        // The String method `substring(a,b)` returns the substring
        // specified by the half-open interval [a,b)

        String firstPart = hello.substring(0,5);
        System.out.println("firstPart:" + firstPart);

        // hello.substring(a) is shorthand for hello.substring(a,
        // hello.length())

        String secondPart = hello.substring(5);
        System.out.println("secondPart:" + secondPart);
    }
}
```

Output:

```
hello: HelloWorld
hello.length(): 10
hello.charAt(0): H
hello.charAt(1): e
hello.charAt(2): l
firstPart:Hello
secondPart:World
```

HelloMath

```
//
// HelloMath.java
//



public class HelloMath
{
    public static void main(String[] args)
    {
        // The Math library defines common mathematical constants
        // and functions.  They are all defined as static members
        // and methods of the Math class.

        System.out.println("pi: " + Math.PI);
        System.out.println("e: " + Math.E);

        System.out.println("cos(0): " + Math.cos(0));
        System.out.println("sin(0): " + Math.sin(0));
        System.out.println("cos(pi): " + Math.cos(Math.PI));
        System.out.println("sin(pi): " + Math.sin(Math.PI));

        // Math.abs() is useful for fuzzy unit tests, which allow
        // for roundoff error in floating point calculations.

        double result = Math.sin(Math.PI);

        if (Math.abs(result) < 1e-6)
            System.out.println("Yippee!");
        else
            System.out.println("Bummer.");
    }
}
```

Output:

```
pi: 3.141592653589793
e: 2.718281828459045
cos(0): 1.0
```

```
sin(0): 0.0
cos(pi): -1.0
sin(pi): 1.2246467991473532E-16
Yippee!
```

HelloRandom

```
//
// HelloRandom.java
//


public class HelloRandom
{
    public static void main(String[] args)
    {
        // Math.random() returns a double in [0,1)

        System.out.println("Random doubles in [0,1):");
        for (int i=0; i<5; i++)
        {
            double value = Math.random();
            System.out.println(value);
        }

        System.out.println();
        System.out.println("Random doubles in [0,10):");
        for (int i=0; i<5; i++)
        {
            double value = Math.random() * 10;
            System.out.println(value);
        }

        System.out.println();
        System.out.println("Random doubles in [200,210):");
        for (int i=0; i<5; i++)
        {
            double value = Math.random() * 10 + 200;
            System.out.println(value);
        }

        System.out.println();
        System.out.println("Random integers in [0,100):");
        for (int i=0; i<5; i++)
        {
            int value = (int)(Math.random() * 100);
```

```
            System.out.println(value);
        }
    }
}
```

Output:

```
Random doubles in [0,1):
0.7014722597039651
0.5201414396696359
0.09984441252727838
0.8753981452285116
0.54847854244902

Random doubles in [0,10):
1.1663874220649961
4.801221245712766
8.452328363574148
9.867377164322509
0.13839014778496028

Random doubles in [200,210):
203.33141093006617
203.72093391403254
206.75863225741566
209.26876107737968
204.76131404962968

Random integers in [0,100):
82
54
4
71
55
```

Coding Exercises: Strings and Math

1. Greetings.

Write a function `greetings()` that takes a single String `name` and returns returns a greeting using the given name. Be sure to include unit tests.

Sample output:

```
greetings("Dr. Kessner") -> "Hello, Dr. Kessner, how are you?"
greetings("Ascii Cat") -> "Hello, Ascii Cat, how are you?"
greetings("Sydneys") -> "Hello, Sydneys, how are you?"
```

2. Attention.

Write a function `attention()` that takes a single `String` as input and returns `true` if the string starts with "Hey, you!". Be sure to include unit tests.

Sample output:

```
attention("Hello, my name is Inigo Montoya.") -> false
attention("Excuse me, Dr. Kessner?") -> false
attention("Hey you! Give me your code!" -> true
```

3. Coin flip.

Write a function that flips a coin randomly, returning a String, either "Heads" or "Tails". Functions involving randomness are a little tricky to write unit tests for. So you should just have your `main()` function print the results from 10 or 20 coin flips to try out your function.

4. Die rolling

Write a function that returns the result of rolling a single 6-sided die. In other words, when you call the function, it should randomly return 1, 2, 3, 4, 5, or 6.

Appendix A: Numeric Conversion

Decimal and Hexadecimal

Decimal is base 10. The digit positions correspond to powers of 10.

$$
\begin{aligned}
1234_{\,\mathrm{DEC}} &= \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline \end{array} \\
&\quad\; 10^3 \;\; 10^2 \;\; 10^1 \;\; 10^0 \\
&= 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0 \\
&= 1000 + 200 + 30 + 4
\end{aligned}
$$

Hexadecimal (hex) is base 16. In hexadecimal we have 16 symbols: the 10 decimal symbols (0-9) and 6 letters (A-F).

decimal	hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	*A*
11	*B*
12	*C*
13	*D*
14	*E*
15	*F*

The digit positions correspond to powers of 16. In code, we write hex numbers with the prefix `0x`. Here are some examples:

$$
\begin{aligned}
0\text{x}10 &= \begin{array}{|c|c|}\hline 1 & 0 \\ \hline \end{array} \\
&\quad\; 16^1 \;\; 16^0 \\
&= 1 \cdot 16^1 + 0 \cdot 16^0 \\
&= 1 \cdot 16 + 0 \cdot 1 \\
&= 16_{\,\mathrm{DEC}}
\end{aligned}
$$

$$
\begin{aligned}
0\text{xA2} &= \underset{16^1}{\boxed{\text{A}}}\,\underset{16^0}{\boxed{2}} \\
&= 10 \cdot 16^1 + 2 \cdot 16^0 \\
&= 10 \cdot 16 + 2 \cdot 1 \\
&= 162_{\,\text{DEC}}
\end{aligned}
$$

$$
\begin{aligned}
0\text{x29A} &= \underset{16^2}{\boxed{2}}\,\underset{16^1}{\boxed{9}}\,\underset{16^0}{\boxed{\text{A}}} \\
&= 2 \cdot 16^2 + 9 \cdot 16^1 + A \cdot 16^0 \\
&= 2 \cdot 256 + 9 \cdot 16 + 10 \cdot 1 \\
&= 666_{\,\text{DEC}}
\end{aligned}
$$

Converting from Decimal to Hex

To convert from decimal to hex, pretend you are making change.

Example To convert 99_{DEC} to hex, we first ask how many 16's it contains. We compute $6 \cdot 16 = 96$, with remainder 3.

$$\begin{aligned} 99_{\text{DEC}} &= 6 \cdot 16 + 3 \\ &= 0\text{x}63 \end{aligned}$$

Example To convert 300_{DEC} to hex, we first ask how many 256's it contains (one, with remainder 44). Then we ask how many 16's are contained in 44 (2, with remainder 12).

$$\begin{aligned} 300_{\text{DEC}} &= 1 \cdot 256 + 2 \cdot 16 + 12 \\ &= 0\text{x}12C \end{aligned}$$

Binary

Binary is base 2, so we have just two symbols: 0 and 1.

decimal	hexadecimal	binary
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	*A*	1010
11	*B*	1011
12	*C*	1100
13	*D*	1101
14	*E*	1110
15	*F*	1111

A binary digit is called a *bit*.

Notice that with 4 bits, we have $2^4 = 16$ possibilities:

$0/1_2$	$0/1_2$	$0/1_2$	$0/1_2$

And with 8 bits, we have $2^8 = 256$ possibilities:

$0/1_2$	$0/1_2$	$0/1_2$	$0/1_2$	$0/1_2$	$0/1_2$	$0/1_2$	$0/1_2$

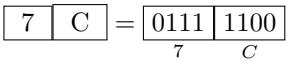
A *byte* is 8 bits (and sometimes 4 bits is called a *nibble*).

We seen that there are 256 possible bytes, corresponding to the decimal values $0-255$, which correspond to the binary values $00000000-11111111$, which correspond to the 2 digit hex values $0\text{x}00 - 0\text{x}FF$.

Converting between Hex and Binary

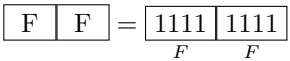
One hex digit corresponds to 4 bits, which makes it easy to convert between hex and binary.

Example



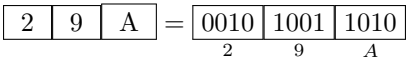
$$0\text{x}7C = 0111\,1100_{\text{BIN}}$$

Example



$$0\text{x}FF = 1111\,1111_{\text{BIN}}$$

Example



$$0\text{x}29A = 0010\,1001\,1010_{\text{BIN}}$$

Octal

Octal is base 8, and we use the eight numeric symbols 0-7.

decimal	hexadecimal	binary	octal
0	0	0	0
1	1	1	1
2	2	10	2
3	3	11	3
4	4	100	4
5	5	101	5
6	6	110	6
7	7	111	7
8	8	1000	10
9	9	1001	11
10	*A*	1010	12
11	*B*	1011	13
12	*C*	1100	14
13	*D*	1101	15
14	*E*	1110	16
15	*F*	1111	17

Example

$$
\begin{aligned}
234_{\,\mathrm{OCT}} &= \underset{8^2}{\boxed{2}}\,\underset{8^1}{\boxed{3}}\,\underset{8^0}{\boxed{4}} \\
&= 2 \cdot 8^2 + 3 \cdot 8^1 + 4 \cdot 8^0 \\
&= 2 \cdot 64 + 3 \cdot 8 + 4 \cdot 1 \\
&= 156_{\,\mathrm{DEC}}
\end{aligned}
$$

One octal digit corresponds to 3 bits ($2^3 = 8$), so conversions between octal and binary are also straightforward.

Example

$$
\boxed{2}\,\boxed{3}\,\boxed{4} = \underset{2}{\boxed{010}}\,\underset{3}{\boxed{011}}\,\underset{4}{\boxed{100}}
$$

$$
234_{\,\mathrm{OCT}} = 010\,011\,100_{\,\mathrm{BIN}}
$$

To convert decimal to octal, pretend you are making change.

Example

To convert 100_{DEC} to octal, we first ask how many 64's it contains (1, with remainder 36). Then we ask how many 8's are in 36 (4, with remainder 4).

$$\begin{aligned} 100_{\,\mathrm{DEC}} &= 1 \cdot 64 + 4 \cdot 8 + 4 \cdot 1 \\ &= 144_{\,\mathrm{OCT}} \end{aligned}$$

Exercises: Numeric Conversion

0) Convert 1234 decimal to hexadecimal. [0x4D2]
1) Convert 0x321 hexadecimal to decimal. [801]
2) Convert 123 decimal to octal. [173]
3) Convert 321 octal to decimal. [209]
4) Convert 123 decimal to binary. [1111011]
5) Convert 123 decimal to binary using your answer from #2.
6) Convert 801 decimal to binary. [1100100001]
7) Convert 801 decimal to binary using your answer from #1.
8) Convert binary 111011001 to octal. [731]
9) Convert binary 11011010 to hex. [0xDA]