Exercises marked with a ⋆ involve writing programs. These exercises do not have accompanying solutions. You are encouraged to work them out on your own. If you need help with any of them, please reach out to the course staff.

Exercise 1. Consider the *j*-- program `$j/j--/tests/SumOfPowers.java`. Assuming you are within the `$j/j--` folder:

a. What is the command to compile the *j*-- compiler?

b. What is the command for just tokenizing the program?

c. What is the command for just parsing the program?

d. What is the command for just pre-analyzing the program?

e. What is the command for just analyzing the program?

f. What is the command for compiling the program into a folder called `out` under `$j/j--`?

g. What is the command for compiling the program into the current folder?

h. What is the command for running `SumOfPowers.class` with command-line arguments 10 and 2?

i. What is the command for decompiling `SumOfPowers.class`, ie, displaying its JVM bytecode instructions?

Exercise 2. Express the following class names in internal form:

a. `Countdown`

b. `jminusminus.Parser`

c. `java.io.File`

d. `java.util.ArrayList`

Exercise 3. Give the field descriptors for the following variable declarations:

a. `boolean w;`

b. `long x;`

c. `java.io.File[] y;`

d. `double[][] z;`

Exercise 4. Give the method descriptors for the following constructor/method declarations:

a. `public Employee(String name) { ... }`

b. `public Coordinate(double latitude, double longitude) { ... }`

c. `public Object get(String key) { ... }`

d. `public void put(String key, Object value) { ... }`

e. `public static void sort(int[] a, Boolean ascending) { ... }`

f. `public static Double[][] transpose(double[][] a) { ... }`

Exercise 5. Consider the following *j*-- program `Locals.java`:

```
import java.lang.System;

public class Locals {
    public static int f(int x, int y) {
        int z = x * x + 2 * x * y + y * y;
        return z;
    }

    public int g(int w, int x) {
        int y = 2 * w * x;
        int z = w * w - y + x * x;
        return z;
    }

    public static void main(String[] args) {
        System.out.println(Locals.f(3, 5));
        System.out.println((new Locals()).g(3, 5));
    }
}
```

a. What are the values for `stack`, `locals`, and `arg_size` for the static method `f()`?

b. What are the stack-frame offsets of the variables (parameters and locals) in the static method `f()`?

c. What are the values for `stack`, `locals`, and `arg_size` for the instance method `g()`?

d. What are the stack-frame offsets of the variables (parameters and locals) in the instance method `g()`?

Exercise 6. Consider the following JVM bytecode:

```
private static int mystery(int);
   0: iconst_1
   1: istore_1
   2: iconst_0
   3: istore_2
   4: iload_1
   5: iload_0
   6: if_icmpgt     21
   9: iload_2
  10: iload_1
  11: iload_1
  12: imul
  13: iadd
  14: istore_2
  15: iinc          1, 1
  18: goto          4
  21: iload_2
  22: ireturn
```

a. What does the method call `mystery(10)` return?

b. What does the method call `mystery(n)` return in general?

Exercise 7 ($\star$)**.** Write a *j*-- program called `Countdown.java` that receives `n` (int) as command-line input and writes as standard output a count from `n` down to 0, both inclusive.

```
$ ./bin/j-- Countdown.java
$ java Countdown 5
5
4
3
2
1
0
```

Exercise 8 ($\star$)**.** Write a *j*-- program called `IsPrime.java` that receives `n` (int) as command-line input and writes as standard output whether `n` is prime or not.

```
$ ./bin/j-- IsPrime.java
$ java IsPrime 31
true
$ java IsPrime 42
false
```

Exercise 9 ($\star$)**.** Write a program called `GenCountdown.java` that uses the `CLEmitter` interface to generate a program called `Countdown.class` that behaves exactly like the *j*-- program described in Exercise 7.

```
$ ./bin/clemitter GenCountdown.java
$ java Countdown 5
5
4
3
2
1
0
```

Exercise 10 ($\star$)**.** Write a program called `GenIsPrime.java` that uses the `CLEmitter` interface to generate a program called `IsPrime.class` that behaves exactly like the *j*-- program described in Exercise 8.

```
$ ./bin/clemitter GenIsPrime.java
$ java IsPrime 31
true
$ java IsPrime 42
false
```

Solutions

Solution 1.

a. `ant`

b. `./bin/j-- -t tests/SumOfPowers.java`

c. `./bin/j-- -p tests/SumOfPowers.java`

d. `./bin/j-- -pa tests/SumOfPowers.java`

e. `./bin/j-- -a tests/SumOfPowers.java`

f. `./bin/j-- -d out tests/SumOfPowers.java`

g. `./bin/j-- tests/SumOfPowers.java`

h. `java SumOfPowers 10 2`

i. `javap -p -v SumOfPowers`

Solution 2.

a. `Countdown`

b. `jminusminus/Parser`

c. `java/io/File`

d. `java/util/ArrayList`

Solution 3.

a. `Z`

b. `J`

c. `[Ljava/io/File;`

d. `[[D`

Solution 4.

a. `(Ljava/lang/String;)V`

b. `(DD)V`

c. `(Ljava/lang/String;)Ljava/lang/Object;`

d. `(Ljava/lang/String;Ljava/lang/Object;)V`

e. `([ILjava/lang/Boolean;)V`

f. `([[D)[[Ljava/lang/Double;`

Solution 5.

a. `stack = 3`, `locals = 3`, `arg_size = 2`

b. `x: 0, y: 1, z: 2`

c. `stack = 3`, `locals = 5`, `arg_size = 3`

d. `this: 0, w: 1, x: 2, y: 3, z: 4`

Solution 6.

a. `385`

b. $1^2 + 2^2 + 3^2 + \cdots + (\mathtt{n} - 1)^2 + \mathtt{n}^2$

Solution 7. Discuss with course staff.

Solution 8. Discuss with course staff.

Solution 9. Discuss with course staff.

Solution 10. Discuss with course staff.