

EXERCISES

1.2.1 Write a `Point2D` client that takes an integer value N from the command line, generates N random points in the unit square, and computes the distance separating the *closest pair* of points.

1.2.2 Write an `Interval1D` client that takes an `int` value N as command-line argument, reads N intervals (each defined by a pair of `double` values) from standard input, and prints all pairs that intersect.

1.2.3 Write an `Interval2D` client that takes command-line arguments N , min , and max and generates N random 2D intervals whose width and height are uniformly distributed between min and max in the unit square. Draw them on `StdDraw` and print the number of pairs of intervals that intersect and the number of intervals that are contained in one another.

1.2.4 What does the following code fragment print?

```
String string1 = "hello";
String string2 = string1;
string1 = "world";
StdOut.println(string1);
StdOut.println(string2);
```

1.2.5 What does the following code fragment print?

```
String s = "Hello World";
s.toUpperCase();
s.substring(6, 11);
StdOut.println(s);
```

Answer: "Hello World". `String` objects are immutable—string methods return a new `String` object with the appropriate value (but they do not change the value of the object that was used to invoke them). This code ignores the objects returned and just prints the original string. To print "WORLD", use `s = s.toUpperCase()` and `s = s.substring(6, 11)`.

1.2.6 A string s is a *circular shift* of a string t if it matches when the characters are circularly shifted by any number of positions; e.g., ACTGACG is a circular shift of TGACGAC, and vice versa. Detecting this condition is important in the study of genomic sequences. Write a program that checks whether two given strings s and t are circular shifts of

one another. *Hint:* The solution is a one-liner with `indexOf()`, `length()`, and string concatenation.

1.2.7 What does the following recursive function return?

```
public static String mystery(String s)
{
    int N = s.length();
    if (N <= 1) return s;
    String a = s.substring(0, N/2);
    String b = s.substring(N/2, N);
    return mystery(b) + mystery(a);
}
```

1.2.8 Suppose that `a[]` and `b[]` are each integer arrays consisting of millions of integers. What does the follow code do? Is it reasonably efficient?

```
int[] t = a; a = b; b = t;
```

Answer. It swaps them. It could hardly be more efficient because it does so by copying references, so that it is not necessary to copy millions of elements.

1.2.9 Instrument `BinarySearch` (page 47) to use a `Counter` to count the total number of keys examined during all searches and then print the total after all searches are complete. *Hint:* Create a `Counter` in `main()` and pass it as an argument to `rank()`.

1.2.10 Develop a class `VisualCounter` that allows both increment and decrement operations. Take two arguments `N` and `max` in the constructor, where `N` specifies the maximum number of operations and `max` specifies the maximum absolute value for the counter. As a side effect, create a plot showing the value of the counter each time its tally changes.

1.2.11 Develop an implementation `SmartDate` of our `Date` API that raises an exception if the date is not legal.

1.2.12 Add a method `dayOfTheWeek()` to `SmartDate` that returns a `String` value `Monday`, `Tuesday`, `Wednesday`, `Thursday`, `Friday`, `Saturday`, or `Sunday`, giving the appropriate day of the week for the date. You may assume that the date is in the 21st century.

EXERCISES *(continued)*

1.2.13 Using our implementation of `Date` as a model (page 91), develop implementations of `Customer` and for `Transaction`. For `Transaction`, use an `int` to represent the amount, in pennies.

1.2.14 Using our implementation of `Date.equals()` as a model (page 103), develop implementations of `equals()` for `Customer` and for `Transaction`.