Data Structures and Algorithms in Java

Assignment 1 (Simple Programs) Discussion

Problem 1 (Greet Three)

`GreetThree.java`

- Command-line input: $name_1$ (String), $name_2$ (String), and $name_3$ (String)
- Standard output: a message containing $name_1$, $name_2$, and $name_3$

```
× ~/workspace/simple_programs
$ javac -d out src/GreetThree.java
$ java GreetThree Alice Bob Carol
Hi Carol, Bob, and Alice.
$ java GreetThree Dan Eve Fred
Hi Fred, Eve, and Dan.
```

Problem 1 (Greet Three)

```
Receive $name_1$ (String), $name_2$ (String), and $name_3$ (String) as command-line inputs

Set $message$ (String) to the value “Hi $name_3$, $name_2$, and $name_1$.”

Write $message$
```

Problem 2 (Three Sort)

ThreeSort.java

- Command-line input: *x* (int), *y* (int), and *z* (int)
- Standard output: the numbers in sorted order

```
× ~/workspace/simple_programs
```

```
$ javac -d out src/ThreeSort.java
$ java ThreeSort 1 3 2
1 2 3
$ java ThreeSort 3 2 1
1 2 3
```

Problem 2 (Three Sort)

Receive *x* (int), *y* (int), and *z* (int) as command-line inputs

Set *alpha* (int) to the smallest of the three numbers

Set *omega* (int) to the largest of the three numbers

Set *middle* (int) to the middle value obtained as an arithmetic combination of *x*, *y*, *z*, *alpha*, and *omega*

Write "*alpha middle omega*"

Problem 3 (Great Circle Distance)

GreatCircle.java

- Command-line input: x_1 (double), y_1 (double), x_2 (double), and y_2 (double)
- Standard output: great circle distance

```
× ~/workspace/simple_programs
$ javac -d out src/GreatCircle.java
$ java GreatCircle 48.87 -2.33 37.8 -122.4
8701.387455462233
$ java GreatCircle 46.36 -71.06 39.90 116.41
10376.503884802196
```

Problem 3 (Great Circle Distance)

Receive x_1 (double), y_1 (double), x_2 (double), and y_2 (double) as command-line inputs

Set d (double) to the great circle distance value computed as

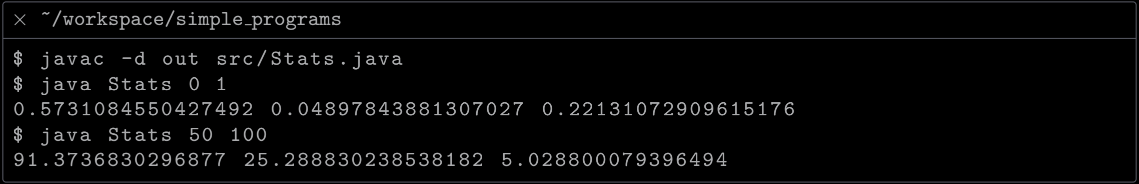
$$d = 6359.83 \arccos(\sin(x_1)\sin(x_2) + \cos(x_1)\cos(x_2)\cos(y_1 - y_2))$$

Write d

Problem 4 (Uniform Random Numbers)

`Stats.java`

- Command-line input: *a* (int) and *b* (int)
- Standard output: mean, variance, and std. deviation of three random numbers drawn from the interval $[a, b)$



```
$ javac -d out src/Stats.java
$ java Stats 0 1
0.5731084550427492 0.048978438813O7027 0.22131072909615176
$ java Stats 50 100
91.3736830296877 25.288830238538182 5.028800079396494
```

Problem 4 (Uniform Random Numbers)

Receive a (int) and b (int) as command-line inputs

Set x_1 (double), x_2 (double), and x_3 (double) to random numbers drawn from the interval $[a, b)$

Set μ (double), var (double), and σ (double) to the mean, variance, and std. deviation values computed as

$$\mu = (x_1 + x_2 + x_3)/3, var = ((x_1 - \mu)^2 + (x_2 - \mu)^2 + (x_3 - \mu)^2)/3, \text{ and } \sigma = \sqrt{var}$$

Write "μ var σ"

Problem 5 (Triangle Inequality)

`Triangle.java`

- Command-line input: x (int), y (int), and z (int)
- Standard output: `true` if each input is less than or equal to the sum of the other two, and `false` otherwise

× ~/workspace/simple_programs

```
$ javac -d out src/Triangle.java
$ java Triangle 3 3 3
true
$ java Triangle 2 4 7
false
```

Problem 5 (Triangle Inequality)

Receive x (int), y (int), and z (int) as command-line inputs

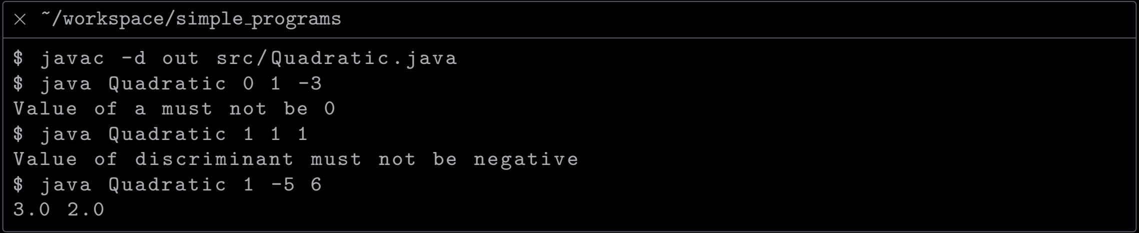
Set *expr* (boolean) to a boolean expression which is `true` if each of x, y, and z is less than or equal to the sum of the other two, and `false` otherwise

Write *expr*

Problem 6 (Quadratic Equation)

`Quadratic.java`

- Command-line input: a (double), b (double), and c (double)
- Standard output: roots of the quadratic equation $ax^2 + bx + c = 0$



```
~/workspace/simple_programs
```

```
$ javac -d out src/Quadratic.java
$ java Quadratic 0 1 -3
Value of a must not be 0
$ java Quadratic 1 1 1
Value of discriminant must not be negative
$ java Quadratic 1 -5 6
3.0 2.0
```

Problem 6 (Quadratic Equation)

Receive a (double), b (double), and c (double) as command-line inputs

If $a = 0$, write the message “Value of a must not be 0”

Otherwise, set *discriminant* (double) to $b^2 - 4ac$

If $discriminant < 0$, write the message “Value of discriminant must not be negative”

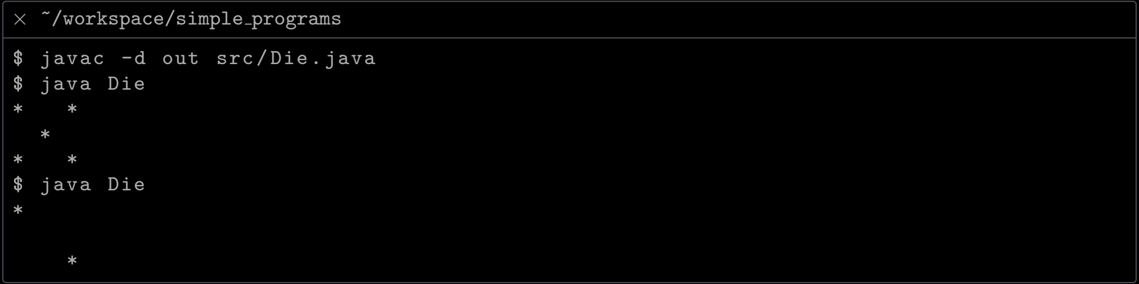
Otherwise, set $root_1$ (double) to $\frac{-b+\sqrt{discriminant}}{2a}$ and $root_2$ (double) to $\frac{-b-\sqrt{discriminant}}{2a}$

Write “$root_1$ $root_2$”

Problem 7 (Six-sided Die)

Die.java

- Standard output: the roll of a six-sided die



~/workspace/simple_programs

```
$ javac -d out src/Die.java
$ java Die
*   *
  *
*   *
$ java Die
*

    *
```

Problem 7 (Six-sided Die)

Set *value* (int) to a random integer from $[1, 6]$

Set *output* (String) to an appropriate string based on *value*

The string format is `".....\n.....\n....."`, where each `.` is either a space or a `*`

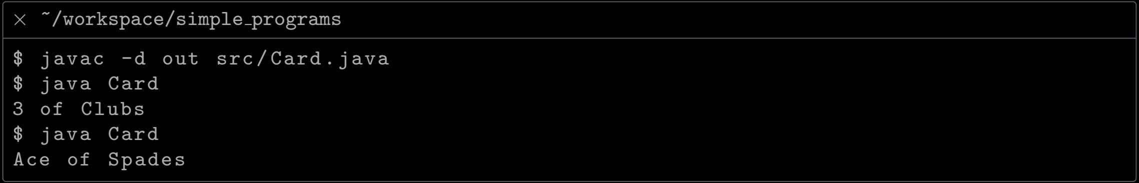
For example, if *value* = 6, the string should be `"* * *\n     \n* * *"`

Write *output*

Problem 8 (Playing Card)

Card.java

- Standard output: a random card from a standard deck of 52 playing cards



```
$ javac -d out src/Card.java
$ java Card
3 of Clubs
$ java Card
Ace of Spades
```

Problem 8 (Playing Card)

Set *rank* (int) to a random integer from $[2, 14]$

Set *rankStr* (String) to a string corresponding to *rank* — the ranks are 2, 3, . . . , *Jack*, *Queen*, *King*, and *Ace*

Set *suit* (int) to a random integer from $[1, 4]$

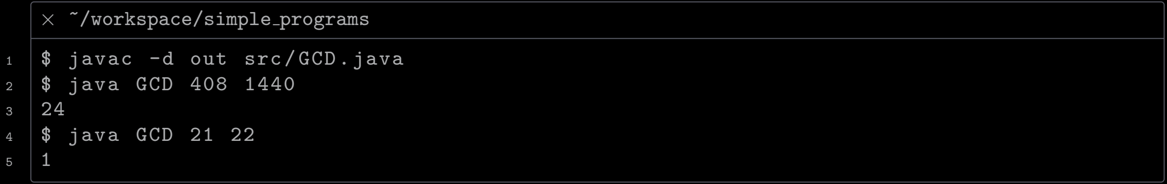
Set *suitStr* (String) to a string corresponding to *suit* — the suits are *Clubs*, *Diamonds*, *Hearts*, and *Spades*

Write "*rankStr* of *suitStr*"

Problem 9 (Greatest Common Divisor)

GCD.java

- Command-line input: p (int) and q (int)
- Standard output: greatest common divisor (GCD) of p and q



```
$ javac -d out src/GCD.java
$ java GCD 408 1440
24
$ java GCD 21 22
1
```

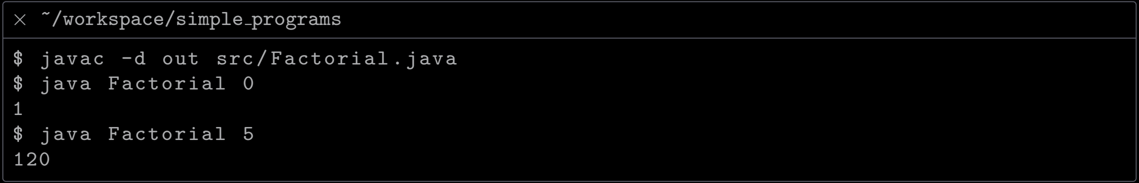
Problem 9 (Greatest Common Divisor)

```
Receive $p$ (int) and $q$ (int) as command-line inputs

Repeat as long as $p \bmod q \neq 0$
    - Exchange $p$ with $q$ and $q$ with $p \bmod q$

Write $q$
```

Problem 10 (Factorial Function)

`Factorial.java`

- Command-line input: *n* (int)
- Standard output: *n*!



```
× ~/workspace/simple_programs
```

```
$ javac -d out src/Factorial.java
$ java Factorial 0
1
$ java Factorial 5
120
```

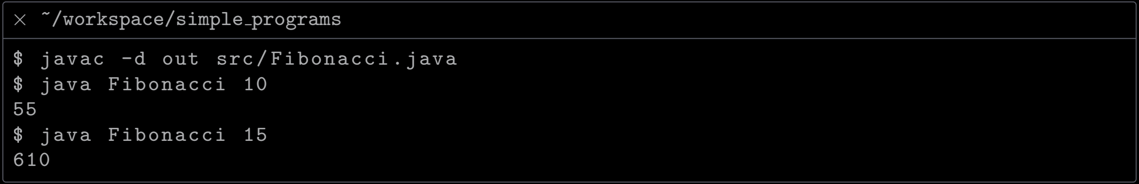
Problem 10 (Factorial Function)

```
Receive $n$ (int) as command-line input

Set $result$ (long) to 1

For each int $i \in [1, n]$
    - Set $result$ to $result * i$

Write $result$
```

Problem 11 (Fibonacci Function)

`Fibonacci.java`

- Command-line input: n (int)
- Standard output: the nth number from the Fibonacci sequence $(0, 1, 1, 2, 3, 5, 8, 13, \ldots)$



```
× ~/workspace/simple_programs
```

```
$ javac -d out src/Fibonacci.java
$ java Fibonacci 10
55
$ java Fibonacci 15
610
```

Problem 11 (Fibonacci Function)

```
Receive $n$ (int) as command-line input

Set $a$ (long) to -1, $b$ (long) to 1, and $i$ (int) to 0

Repeat as long as $i \leq n$
    - Exchange $a$ with $b$ and $b$ with $a + b$
    - Increment $i$ by 1

Write $b$
```

Problem 12 (Primality Test)

`PrimalityTest.java`

- Command-line input: *n*
- Standard output: *true* if *n* is prime, and *false* otherwise

```
× ~/workspace/simple_programs
```

```
$ javac -d out src/PrimalityTest.java
$ java PrimalityTest 31
true
$ java PrimalityTest 42
false
```

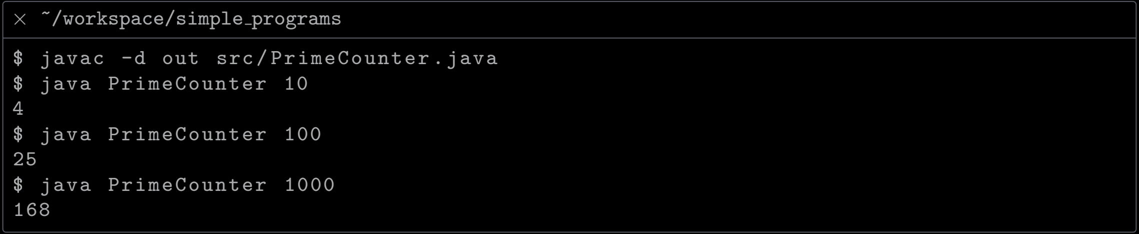
Problem 12 (Primality Test)

```
Receive $n$ (int) as command-line input

Set $i$ (int) to 2

Repeat as long as $i \leq n/i$
    - If $i$ divides $n$, break
    - Increment $i$ by 1

If $i > n/i$, write true; otherwise, write false
```

Problem 13 (Counting Primes)

`PrimeCounter.java`

- Command-line input: *n* (int)
- Standard output: number of primes less than or equal to *n*



```
$ javac -d out src/PrimeCounter.java
$ java PrimeCounter 10
4
$ java PrimeCounter 100
25
$ java PrimeCounter 1000
168
```

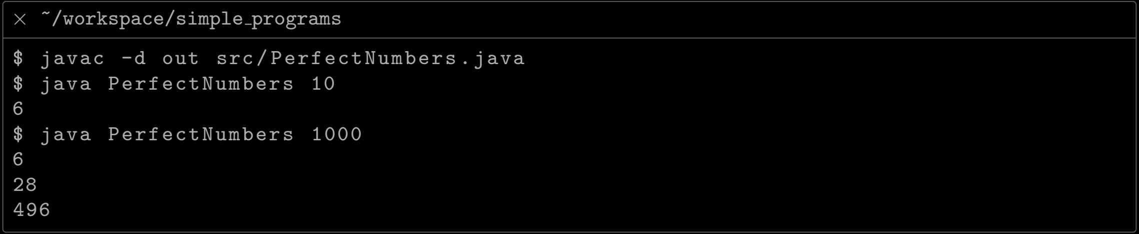
Problem 13 (Counting Primes)

```
Receive $n$ (int) as command-line input

Set $count$ (int) to 0

For each int $i \in [2, n]$
    - Set $j$ (int) to 2
    - Repeat as long as $j \leq i/j$
        - If $j$ divides $i$, break
        - Increment $j$ by 1
    - If $j > i/j$, increment $count$ by 1

Write $count$
```

Problem 14 (Perfect Numbers)

`PerfectNumbers.java`

- Command-line input: *n* (int)
- Standard output: perfect numbers that are less than or equal to *n*



× ~/workspace/simple_programs

```
$ javac -d out src/PerfectNumbers.java
$ java PerfectNumbers 10
6
$ java PerfectNumbers 1000
6
28
496
```

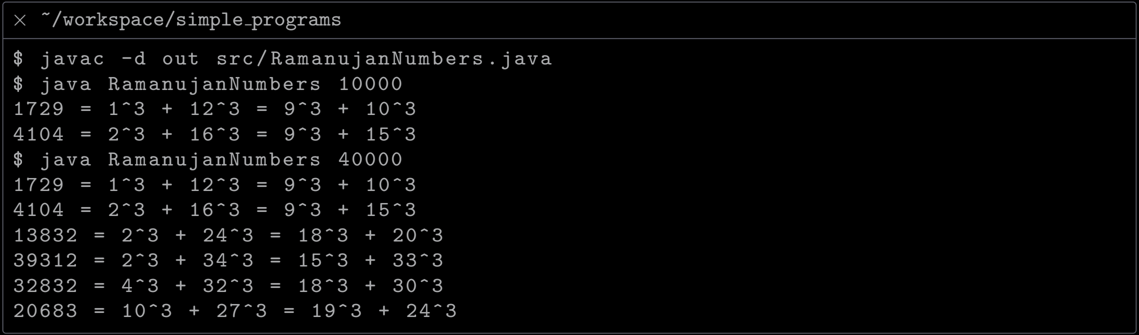
Problem 14 (Perfect Numbers)

```
Receive $n$ (int) as command-line input

For each int $i \in [2, n]$
    - Set $total$ (int) to 0
    - For each int $j \in [1, i/2]$
        - If $j$ divides $i$, increment $total$ by $j$
    - If $total = i$, write $i$
```

Problem 15 (Ramanujan Numbers)

RamanujanNumbers.java

- Command-line input: *n* (int)
- Standard output: integers $\leq n$ that can be expressed as the sum of two cubes in two different ways



```
$ javac -d out src/RamanujanNumbers.java
$ java RamanujanNumbers 10000
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
$ java RamanujanNumbers 40000
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
13832 = 2^3 + 24^3 = 18^3 + 20^3
39312 = 2^3 + 34^3 = 15^3 + 33^3
32832 = 4^3 + 32^3 = 18^3 + 30^3
20683 = 10^3 + 27^3 = 19^3 + 24^3
```

Problem 15 (Ramanujan Numbers)

```
Receive $n$ (int) as command-line input

Set $a$ (int) to 1

Repeat as long as $a^3 \leq n$
  - Set $b$ (int) to $a + 1$
  - Repeat as long as $a^3 + b^3 \leq n$
      - Set $c$ (int) to $a + 1$
      - Repeat as long as $c^3 \leq n$
          - Set $d$ (int) to $c + 1$
          - Repeat as long as $c^3 + d^3 \leq n$
              - Set $x$ (int) to $a^3 + b^3$ and $y$ (int) to $c^3 + d^3$
              - If $x = y$, write "$x = a$^3 $+ b$^3 $= c$^3 $+ d$^3"
              - Increment $d$ by 1
          - Increment $c$ by 1
      - Increment $b$ by 1
  - Increment $a$ by 1
```