Objectives. Implement simple data types that:

1. Conform to a given API.
2. Are immutable.
3. Override methods `equals()` and `toString()` from `Object`.

Problem 1. (*Six-sided Die*) Implement a data type `Die` in `Die.java` that represents a six-sided die and supports the following API:

method	description
`Die()`	construct a die
`void roll()`	roll the die
`int value()`	face value of the die
`boolean equals(Die that)`	does the die have the same face value as *that*?
`String toString()`	a string representation of the current face value of the die, ie, ⚀, ⚁, ⚂, ⚃, ⚄, or ⚅

```
$ java Die 5 3 3
*   *
  *
*   *
false
true
```

Problem 2. (*US Phone Number*) Implement an immutable data type `PhoneNumber` in `PhoneNumber.java` that represents a US phone number, and supports the following API:

method	description
`PhoneNumber(int area, int exch, int ext)`	construct a phone number given the area code, exchange, and extension
`boolean equals(PhoneNumber that)`	is the phone number same as *that*?
`String toString()`	a string representation of the phone number, in `"(area) exch-ext"` format (use `String.format()`)

```
$ java PhoneNumber
(609) 258-4455
(609) 876-5309
(609) 003-5309
(215) 876-5309
(609) 876-5309
true
false
true
true
```

Problem 3. (*Geo Location*) Implement an immutable data type `Location` in `Location.java` that represents a location on Earth and supports the following API:

method	description
`Location(String loc, double lat, double lon)`	construct a new location given its name, latitude, and longitude values
`double distanceTo(Location that)`	the great-circle distance[†] between this location and *that*
`boolean equals(Location that)`	is this location the same as *that*?
`String toString()`	a string representation of the location, in `"loc (lat, lon)"` format

† See Problem 4 of Homework 1 for formula

```
$ java Location 5 40.6769 117.2319
The Great Wall of China (China) (40.6769, 117.2319)
Petra (Jordan) (30.3286, 35.4419)
The Colosseum (Italy) (41.8902, 12.4923)
Chichen Itza (Mexico) (20.6829, -88.5686)
Machu Picchu (Peru) (-13.1633, -72.5456)
Taj Mahal (India) (27.175, 78.0419)
Christ the Redeemer (Brazil) (22.9519, -43.2106)
3868.964067791193
false
```

Problem 4. (*3D Point*) Implement an immutable data type `Point3D` in `Point3D.java` that represents a point in 3D and supports the following API:

method/class	description
`Point3D(double x, double y, double z)`	construct a point in 3D given its coordinates
`double distance(Point3D that)`	the Euclidean distance† between this point and *that*
`String toString()`	a string representation of the point, in `"(x, y, z)"` format

† The Euclidean distance between the points (x_1, y_1, z_1) and (x_2, y_2, z_2) is given by $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$

```
$ java Point3D
3
-3 1 6
0 5 8
-5 -7 -3
(-3.0, 1.0, 6.0), distance to origin = 6.782329983125268
(0.0, 5.0, 8.0), distance to origin = 9.433981132056603
(-5.0, -7.0, -3.0), distance to origin = 9.1104335791443
```

Problem 5. (*Rational Number*) Implement a data type `Rational` in `Rational.java` that represents a rational number, ie, a number of the form a/b where a and $b \neq 0$ are integers. The data type must support the following API:

method	description
`Rational(long x)`	construct a rational number whose numerator is the given number and denominator is 1
`Rational(long x, long y)`	construct a rational number given its numerator and denominator†
`Rational add(Rational that)`	the sum of this and *that* rational number
`Rational multiply(Rational that)`	the product of this and *that* rational number
`String toString()`	a string representation of the rational number

† Use the private method `gcd()` to ensure that the numerator and denominator never have any common factors. For example, the rational number 2/4 must be represented as 1/2.

```
$ java Rational 10
1023/512
```

Files to Submit

1. `Die.java`
2. `PhoneNumber.java`
3. `Location.java`
4. `Point3D.java`
5. `Rational.java`

Before you submit:

- Make sure your programs meet the input and output specifications by running the following command on the terminal:

  ```
  $ python3 run_tests.py -v [<problems>]
  ```

  where the optional argument `<problems>` lists the problems (`Problem1`, `Problem2`, etc.) you want to test, separated by spaces; all the problems are tested if no argument is given.

- Make sure your programs meet the style requirements by running the following command on the terminal:

  ```
  $ check_style <program>
  ```

  where `<program>` is the `.java` file whose style you want to check.