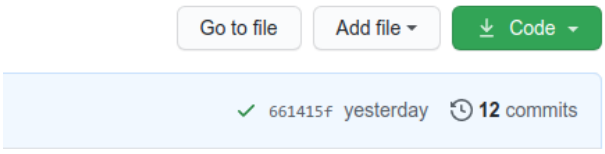
Atividade 02
Objetos e classes
Prof. Diogo S. Martins

Orientações gerais

- **URL GitHub Classroom:** https://classroom.github.com/a/JcLvoUKF
- O projeto inicialmente possui erros de compilação pois estão faltando as classes solicitadas nos enunciados. Portanto, antes de executar os testes, garanta que definiu as classes e seus respectivos métodos, mesmo que os conteúdos dos métodos sejam vazios ou sem efeito.
- Todas as classes devem ser criadas na pasta `src/main/java`. Tanto as classes, quanto as assinaturas dos métodos, devem ter nomes idênticos ao que está especificado no enunciado, caso contrário os testes falharão.
- Não altere as classes da pasta `src/test/java`. Qualquer alteração nestas classes será ignorada no momento da correção pelo professor, pois os arquivos de teste serão substituídos pelas versões originais.
- O programa somente será aceito pelo sistema de submissão quando passar em todos os testes. Portanto, lembre-se de executar a operação "Gradle check", conforme orientado na aula.

O que entregar

1. **No GitHub Classroom**. Completar a atividade, de acordo com as questões enunciadas, e submeter (`push`).
2. **No Moodle**. Colar o código hash do último commit efetuado (essa é a versão que o professor irá corrigir). A figura abaixo exemplifica onde encontrar a hash do último commit na página do seu repositório (a hash nesse exemplo é `661415f`).



Questões

1. **Racionais**. A classe `Rational` implementa um tipo de dados que representa números racionais imutáveis. Implemente os métodos solicitados.

   - `public Rational (int numerator, int denominator)`
     Construtor.
     Por exemplo, para criar o racional $\frac{3}{5}$:

```
1 Rational r = new Rational(3, 5)
```

- `public String toString()`

  Converte o número racional para `String`, no formato `"<numerator>/<denominator>"`. Caso o número seja negativo, o sinal deve aparecer na frente do numerador. Exemplo:

```
Rational a = new Rational(1, 2);
Rational b = new Rational(-1, 3);
Rational c = new Rational(1, -3);
Rational d = new Rational(-1, -2);

System.out.printf("%s %s %s %s\n", a, b, c, d);
```

```
1/2 -1/3 -1/3 1/2
```

- `public Rational plus(Rational b)`

  Adição entre dois números racionais.

  Por exemplo, para a operação $\frac{1}{2} + \frac{3}{5}$:

```
Rational a = new Rational(1, 2);
Rational b = new Rational(3, 5);
Rational c = a.plus(b);

System.out.printf("%s + %s = %s\n", a, b, c);
```

```
1/2 + 3/5 = 11/10
```

- `public Rational minus(Rational b)`

  Subtração entre dois racionais.

  Por exemplo, para a operação $\frac{1}{2} - \frac{3}{5}$:

```
Rational a = new Rational(1, 2);
Rational b = new Rational(3, 5);
Rational c = a.minus(b);

System.out.printf("%s - %s = %s\n", a, b, c);
```

```
1/2 - 3/5 = -1/10
```

- `public Rational times(Rational b)`

  Multiplicação entre dois racionais.

  Por exemplo, para a operação $\frac{1}{2} \times \frac{3}{5}$:

```
Rational a = new Rational(1, 2);
Rational b = new Rational(3, 5);
Rational c = a.times(b);

System.out.printf("%s * %s = %s\n", a, b, c);
```

```
1/2 * 3/5 = 3/10
```

- `public Rational divides(Rational b)`

  Divisão entre dois racionais.

  Por exemplo, para a operação $\frac{1}{2} \div \frac{3}{5}$, fazemos:

```
Rational a = new Rational(1, 2);
Rational b = new Rational(3, 5);
Rational c = a.divides(b);

System.out.printf("(%s) / (%s) = %s\n", a, b, c);
```

```
(1/2) / (3/5) = 5/6
```

- `public boolean equals(Rational b)`

  Igualdade entre dois racionais. Exemplo:

```
Rational a = new Rational(1, 2);
Rational b = new Rational(1, 2);
Rational c = new Rational(1, 3);

boolean d = a.equals(b);
boolean e = a.equals(c);

System.out.printf("%s equals %s: %s\n", a, b, d);
System.out.printf("%s equals %s: %s\n", a, c, e);
```

```
1/2 equals 1/2: true
1/2 equals 1/3: false
```

Observações:

- Caso necessário, implemente métodos auxiliares para calcular o máximo divisor comum (mdc[1]) e o mínimo múltiplo comum (mmc[2]).
- Dois números racionais são considerados iguais se suas formas simplificadas são idênticas. Por exemplo, $\frac{12}{36}$ e $\frac{1}{3}$ são iguais, pois $\frac{12}{36}$ pode ser simplificado para $\frac{1}{3}$. Portanto, é essencial que os números sejam simplificados durante a comparação no método `equals`.

2. **Matrizes.** A classe `Matrix` abstrai, como objetos imutáveis, matrizes de números reais, com quantidades arbitrárias de linhas e de colunas. Os métodos da classe implementam as principais operações aritméticas e relacionais entre instâncias de matrizes. Implemente os métodos solicitados.

   - `public Matrix(double[][] cells)`

     Construtor. O argumento é um array bidimensional contendo os valores da matriz. Por exemplo, considere:

$$m = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

     Para representar m como um objeto do tipo `Matrix`:

```
1 double[][] cells = new double[][] {
2   {1, 2, 3},
3   {4, 5, 6}
4 };
5 Matrix m = new Matrix(cells);
```

[1] https://en.wikipedia.org/wiki/Euclidean_algorithm#Procedure
[2] https://en.wikipedia.org/wiki/Least_common_multiple#Using_the_greatest_common_divisor

- `public String toString()`

  Converte a matriz para `String`, exibindo-a no formato tabular. Para fins de simplicidade, considere como critérios fixos que cada coluna tem largura de 10 posições e que os números reais devem ser exibidos com 6 casas decimais. Exemplo:

```
Matrix m = new Matrix(new double[][] {
    {1, 2, 3},
    {4, 5, 6}
});
System.out.print(m);
```

```
1.000000  2.000000  3.000000
4.000000  5.000000  6.000000
```

- `public int lines()`

  `public int columns()`

  Retornam, respectivamente, a quantidade de linhas e a quantidade de colunas da matriz.

- `public double get(int line, int column)`

  Retorna o valor da posição identificada pelos parâmetros. Considere que a primeira linha (analogamente, a primeira coluna) seja zero. Caso haja tentativa de acessar uma posição que não exista na matriz, o método deve disparar uma exceção do tipo `IllegalArgumentException`.

- `public Matrix plus(Matrix m)`

  Efetua a adição entre duas matrizes. Caso haja tentativa de adicionar matrizes de dimensões incompatíveis, deve-se disparar uma exceção do tipo `IllegalArgumentException`.

  Suponha a operação:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

  Equivale ao seguinte trecho utilizando `Matrix`:

```
Matrix a = new Matrix(new double[][] {
    {1.0, 2.0},
    {3.0, 4.0}
});
Matrix b = new Matrix(new double[][] {
    {5.0, 6.0},
    {7.0, 8.0}
});
Matrix c = a.plus(b);

System.out.print(c);
```

```
  6.000000  8.000000
 10.000000 12.000000
```

- `public Matrix minus(Matrix m)`

  Efetua a subtração entre duas matrizes. Caso haja tentativa de subtrair matrizes de dimensões incompatíveis, deve-se disparar uma exceção do tipo `IllegalArgumentException`.

```
Matrix a = new Matrix(new double[][] {
    {1.0, 2.0},
    {3.0, 4.0}
});
Matrix b = new Matrix(new double[][] {
    {5.0, 6.0},
    {7.0, 8.0}
});
Matrix c = a.minus(b);

System.out.print(c);
```

```
 -4.000000 -4.000000
 -4.000000 -4.000000
```

- `public Matrix times(double scalar)`

  Efetua a multiplicação entre uma matriz e um escalar. Exemplo:

```
Matrix a = new Matrix(new double[][] {
    {1.0, 2.0},
    {3.0, 4.0}
});
double b = 5.0;
Matrix c = a.times(b);

System.out.print(c);
```

```
  5.000000 10.000000
 15.000000 20.000000
```

- `public Matrix times(Matrix m)`

  Efetua a multiplicação[3] entre duas matrizes. Caso haja tentativa de multiplicar matrizes de dimensões incompatíveis, deve-se disparar uma exceção do tipo `IllegalArgumentException`. Exemplo:

```
Matrix a = new Matrix(new double[][] {
    {1.0, 2.0, 3.0},
    {3.0, 4.0, 5.0}
});
Matrix b = new Matrix(new double[][] {
    {1.0, 2.0},
    {3.0, 4.0},
    {5.0, 6.0}
});
Matrix c = a.times(b);

System.out.print(c);
```

```
22.000000 28.000000
40.000000 52.000000
```

- `public Matrix transpose()`

  Retorna a transposta[4] de uma matriz. Exemplo:

[3] https://www.mathsisfun.com/algebra/matrix-multiplying.html

[4] https://en.wikipedia.org/wiki/Transpose

```
Matrix a = new Matrix(new double[][] {
    {1.0, 2.0, 3.0},
    {3.0, 4.0, 5.0}
});
Matrix b = a.transpose();

System.out.print(b);
```

```
  1.000000  3.000000
  2.000000  4.000000
  3.000000  5.000000
```

- `public boolean isSquare()`

  Verifica se a matriz é quadrada. Exemplo:

```
Matrix a = new Matrix(new double[][] {
    {1.0, 2.0, 3.0},
    {3.0, 4.0, 5.0}
});
Matrix b = new Matrix(new double[][] {
    {5.0, 6.0},
    {7.0, 8.0}
});

System.out.println(a.isSquare());
System.out.println(b.isSquare());
```

```
false
true
```

- `public boolean isSymmetric()`

  Verifica se a matriz é simétrica[5].

```
Matrix a = new Matrix(new double[][] {
    {1, 7, 3},
    {7, 4, 5},
    {3, 5, 0}
});
Matrix b = new Matrix(new double[][] {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
});

System.out.println(a.isSymmetric());
System.out.println(b.isSymmetric());
```

```
true
false
```

[5] https://en.wikipedia.org/wiki/Symmetric_matrix