Quando se trata de gerar estatísticas em Java, existem classes que podem simplificar bastante o trabalho. Duas delas são DoubleSummaryStatistics e IntSummaryStatistics, que são usadas para calcular estatísticas básicas em coleções de valores do tipo double e int, respectivamente.

A classe DoubleSummaryStatistics é uma classe utilitária que permite calcular estatísticas como soma, média, valor mínimo e máximo, além do total de elementos em uma coleção de valores do tipo double. Ela possui métodos como getSum(), getAverage(), getMin(), getMax() e getCount(), que retornam os respectivos valores estatísticos.

Da mesma forma, a classe IntSummaryStatistics é usada para calcular estatísticas básicas em coleções de valores do tipo int. Ela também possui métodos como getSum(), getAverage(), getMin(), getMax() e getCount() para obter as estatísticas desejadas.

Essas classes são especialmente úteis quando você precisa realizar cálculos estatísticos em uma coleção de valores. Por exemplo, se você tem uma lista de preços de produtos e deseja calcular a média, o valor mínimo e o valor máximo, você pode utilizar a classe DoubleSummaryStatistics para obter essas informações de forma simples e eficiente, de forma bem similar ao que fizemos ao pegar as estatísticas dos episódios com maior e menor avaliação, a média e o total de itens levados em consideração na montagem de estatísticas.

Para usar essas classes, você pode criar uma instância delas e, em seguida, percorrer a coleção de valores, atualizando as estatísticas conforme necessário. Ao final do processo, você pode chamar os métodos apropriados para obter as estatísticas desejadas. Vamos ver um exemplo onde temos uma lista de estudantes e queremos pegar dados sobre suas idades.

Para a classe Aluno teremos o seguinte código:

**public** **class** **Aluno** {

**private** String nome;

**private** LocalDate nascimento;

**public** **Aluno**(String nome, LocalDate nascimento) {

**this**.nome = nome;

**this**.nascimento = nascimento;

}

**public** int **getIdade**() {

Period periodo = Period.between(nascimento, LocalDate.now());

**return** periodo.getYears();

}

//getters, setters e toString omitidos

}

Criamos um método getIdade(), onde estamos usando o Period, que assim como o LocalDate é uma classe do pacote java.time, para calcular a idade do estudante de acordo com o dia que a aplicação está sendo executada.

Em nossa classe Principal teremos o seguinte código:

**public** **class** **Principal** {

**public** **static** void **main**(String[] args) {

**List**<**Aluno**> alunos = **Arrays**.**asList**(

**new** **Aluno**("Alice", **LocalDate**.**of**(2002, 10, 20)),

**new** **Aluno**("Bob", **LocalDate**.**of**(1980, 8, 9)),

**new** **Aluno**("Carlos", **LocalDate**.**of**(2001, 01, 28)),

**new** **Aluno**("David", **LocalDate**.**of**(2003, 05, 12)),

**new** **Aluno**("Eva", **LocalDate**.**of**(2005, 12, 03))

);

**IntSummaryStatistics** stats = alunos.**stream**()

.**mapToInt**(**Aluno**::getIdade)

.**summaryStatistics**();

// Exibindo as estatísticas

**System**.out.**println**("Idade média: " + stats.**getAverage**());

**System**.out.**println**("Mínima idade: " + stats.**getMin**());

**System**.out.**println**("Máxima idade: " + stats.**getMax**());

**System**.out.**println**("Total de alunos: " + stats.**getCount**());

}

}

OBS: LocalDate.of = O método LocalDate.of em Java é usado para criar uma instância de LocalDate, que representa uma data (ano, mês e dia) no formato ISO-8601 sem fuso horário. A classe LocalDate faz parte do pacote java.time, que foi introduzido no Java 8 como parte da nova API de data e hora.