02. Listas de objetos

Date	@26/10/2022
Categoria	Java
Curso	Java Collections: Dominando Listas Sets e Mapas

Tópicos

- Listas de objetos
- Criando listas a partir de objetos próprios
- Reescrevendo o toString da classe Aula
- Definindo um critério de comparação na classe Aula
- Ordenando com Java 8

Criando listas a partir de objetos próprios

Crie uma lista com os três objetos do tipo `Aula` presentes dentro do método `main` da classe `TestaListaDeAula` e imprima a lista da mesma forma que estávamos fazendo na aula anterior. Qual será o resultado? O nome das 3 aulas?

```
import java.util.*;

public class TestaListaDeAula {

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

Ao executar esse código o resultado será parecido com:

```
[Aula@c3bfe4, Aula@d24512, Aula@c13eaa1]
```

```
        Aula a1 = new Aula("Revistando as ArrayLists", 21);
        Aula a2 = new Aula("Listas de objetos", 20);
        Aula a3 = new Aula("Relacionamento de listas e objetos", 15);

        // codigo para criar a lista de aulas

        ArrayList<Aula> aulas = new ArrayList<>();
        aulas.add(a1);
        aulas.add(a2);
        aulas.add(a3);

        System.out.println(aulas);
    }
}
```

Isso aconteceu porque o método `toString` da classe `ArrayList` percorre todos os elementos da lista, concatenando seus valores também de `toString`. Como a classe `Aula` não possui um `toString` reescrito (override), ele utilizará o `toString` definido em `Object`, que retorna o nome da classe, concatenado com um @ e seguido de um identificador único do objeto.

Definindo um critério de comparação na classe Aula

Diferente de uma `String` ou de tipos primitivos mais simples, o `Collections.sort` não sabe ordenar uma lista de `Aula`. De qual forma ele faria isso? Pelo nome da aula? Pela duração? Não daria para saber. Para que ele seja capaz de fazer isso, você precisa implementar a interface `Comparable` definindo um critério de comparação para os objetos desse tipo.