۱۴.

درهمان پروژه کارگاه سیزدهم:

- می‌خواهیم بتوانیم کارمندان یک شرکت را براساس میزان حقوق ماهانه‌شان مرتب کرد. برای این منظور می‌توان از متد استاتیک `sort` از کلاس `Collections` استفاده کرد (`java.util.Collections`). `API` این متد به شکل زیر است:

`public static void` **`sort`**`(List list)`

Sorts the specified list into ascending order, according to the *natural ordering* of its elements. All elements in the list must implement the `Comparable` interface. Furthermore, all elements in the list must be *mutually comparable* (that is, `e1.compareTo(e2)` must not throw a `ClassCastException` for any elements `e1` and `e2` in the list).

همانطور که از متن مشخص است، المان‌های داخل لیست برای مرتب‌سازی باید قابل مقایسه با هم باشند. برای این منظور این متد از اینترفیس `Comparable` استفاده می‌کند که دارای یک متد با نام `()compareTo` است. در صورت عدم پیاده‌سازی این اینترفیس و استفاده از متد `sort`، استثنای `ClassCastException` پرتاب می‌شود. `API` متد `()compareTo` به شکل زیر است:

۱.

داخل کلاس `Company` متدی با نام `sortEmployees` بنویسید که کارمندان را براساس میزان حقوق دریافتی‌شان مرتب کند.

۲.

- در همین پروژ یک کلاس برای ذخیره سازی اعداد `integer` با نام `Number` بنویسید (درهرشی یک عدد ذخیره می‌شود).
- سیاستی برای جمع دو شی از نوع `Number` اتخاذ کنید.
- سیاستی برای انجام مقایسه بین کلاس‌های `Number` و `Employee` اضافه کنید:
  - دو شی از نوع `Number` با هم برابرند اگر مقادیر دو عدد یکسان باشد.
  - دو شی از نوع `Employee` با هم برابرنداگرحقوق ماهانه آنها برابر باشد.
- یک شی از نوع `Employee` با یک شی از نوع `Number` برابر است اگر مقدار شی از کلاس `Number` برابر با حقوق ماهانه شی از کلاس `Employee` باشد.

بهترین پیاده‌سازی چگونه است؟