ZADANIE

1. Generowanie przykładowych danych (Random, Math)

Napisz metodę, która generuje listę pracowników:

- losuj imię z tablicy imion,
- losuj dział (np. „IT”, „HR”, „Marketing”),
- losuj wiek 20–65,
- losuj pensję od 3500 do 15000,
- oblicz „wskaźnik efektywności” pracownika:

wzór efektywność:

$$efektywność = \frac{\sqrt{wiek} \cdot pensja}{100}$$

---

2. Zapis danych pracownika do struktur (Map, Set)

Dla wygenerowanego pracownika:

- zapisz jego dane w HashMap<String, Object>,
  np. `"name" → "Adam"`, `"salary" → 7200`, `"department" → "IT"`, `"age" → 34`.
- dodaj identyfikator pracownika do HashSet<Integer>,
  aby upewnić się, że ID pracowników są unikalne.

---

3. Praca z datami (java.time)

Do każdego pracownika:

- przypisz datę zatrudnienia (losowo w ostatnich 10 latach),

- wylicz staż pracy w latach na podstawie dzisiejszej daty,
- wyświetl datę najbliższego przeglądu rocznego (zawsze 1 marca następnego roku).

---

4. Sortowanie pracowników (Comparator)

Napisz Comparator, który sortuje pracowników:

1. po wysokości pensji malejąco,
2. jeśli pensja taka sama — po stażu pracy rosnąco.

---

5. Operacje na kolekcjach (Collections)

Wykonaj:

- `Collections.sort(lista, comparator)` – sortowanie pracowników,
- `Collections.shuffle(lista)` – losowe przetasowanie listy,
- `Collections.max(lista, comparator)` – pracownik z najwyższą pensją.

---

6. Zabezpieczenia i porównania (Objects)

W swoim kodzie:

- przed dodaniem pracownika zapewnij:
  `Objects.requireNonNull(employee);`
- przy porównywaniu obiektów użyj `Objects.equals(a, b)`,
- stosuj `Objects.toString(value, "Brak danych")` przy wyświetlaniu.