Time limit 1 s
Memory limit 64 MB

Nama File: OperatingSystem.java, Linux.java, MacOS.java, Windows.java, UsageType.java

Buat sistem penilaian kompatibilitas untuk berbagai *Operating System* dengan menerapkan konsep **inheritance** pada Java. Implementasikan keseluruhan struktur sesuai ketentuan berikut.

Ketentuan Umum

- Seluruh atribut pada setiap kelas harus diberi modifier `private` dan nilainya dapat diakses dengan fungsi getter, contoh: getBaseScore. Penamaan fungsi getter adalah get<Nama Attribute> dengan nama fungsi tersebut dituliskan dalam Camel Case.
- Method `printInfo` dan `calculateCompatibility` wajib `public`.
- Method `calculateCompatibility` dihitung dengan cara baseScore + multiplier * baseScore
- Method bantu pada kelas abstrak, yaitu `clampScore` dan `getAdditionalInfo`, harus menggunakan modifier `protected`.
- Enum, konstruktor, dan getter yang disebutkan pada spesifikasi harus `public`.

1. Kelas Abstrak OperatingSystem

- Atribut: `name` (String), `version` (String), `kernelType` (String), `baseScore` (double 0–100).
- Buat konstruktor dengan dengan parameter name, version, dan kernelType, dan baseScore. Apabila parameter yang diberikan bernilai null, maka akan digantikan dengan default value yaitu secara berurutan: "Unknown OS", "Unknown Version", "Unknown Kernel", dan 0.
- Buat method abstrak `double calculateCompatibility(UsageType usage)`.
- Sediakan method bantu `protected double clampScore(double score)` untuk memastikan skor tetap di rentang 0–100.
- Implementasikan method `void printInfo(UsageType usage)` untuk menampilkan informasi OS beserta nilai kompatibilitas hasil `calculateCompatibility`.
- **Format Input**:
  - Konstruktor menerima parameter `name`, `version`, `kernelType`, dan `baseScore` dalam urutan tersebut. Setiap nilai teks null atau kosong harus diganti dengan default yang sudah ditentukan.
  - Method `calculateCompatibility` dan `printInfo` menerima satu argumen bertipe `UsageType` yang menunjukkan skenario penggunaan yang sedang dinilai.
- **Format Output**:
  - `printInfo` harus mencetak lima baris utama secara berurutan:
    ```
    Name: <name>
    Version: <version>
    Kernel Type: <kernelType>
    {Additional Information jika ada.}
    Base Score: <baseScore dengan 2 digit desimal>
    Compatibility for <usage.name()>: <skor kompatibilitas dengan 2 digit desimal>.
    ```
  - Jika `getAdditionalInfo()` mengembalikan string non-kosong, baris tersebut dicetak setelah informasi kernel dan sebelum baris `Base Score`.
  - Gunakan `System.out.printf` atau formatter lain yang setara sehingga nilai numerik muncul dengan dua angka di belakang koma dan garis baru di akhir setiap baris.

2. Enum UsageType

- Buat enum `UsageType` dengan nilai `SERVER`, `DEVELOPMENT`, dan `GAMING`.

3. Kelas Turunan

- **Linux**
  - Tambahkan enum internal `Distro` dengan nilai `UBUNTU`, `FEDORA`, `ARCH`, `DEBIAN`.

    - Atribut tambahan: `distroType` (Distro). Jika null, gunakan default `UBUNTU`.
    - Aturan multiplier terhadap `baseScore`:
        - `SERVER` → +20%
        - `DEVELOPMENT` → +10%
        - `GAMING` → -15%
    - **Format Input**:
        - Konstruktor menerima parameter tambahan `distroType`. Jika null, setel ke `Distro.UBUNTU`.
        - `getDistroType()` mengembalikan nilai enum untuk kebutuhan pengujian.
    - **Format Output**:
        - `getAdditionalInfo()` harus mengembalikan string `"Distribution: " + distroType.name()`.
        - `printInfo` wajib menampilkan baris tambahan `Distribution: <DISTRO>` tepat sebelum baris `Base Score`, sesuai nilai yang dikembalikan oleh `getAdditionalInfo`.
- **MacOS**
    - Atribut tambahan: `hasMSeriesChip` (boolean).
    - Aturan multiplier baseScore:
        - `DEVELOPMENT` → +25%
        - `GAMING` → -20%
        - Jika `hasMSeriesChip == true` → +10%
    - **Format Input**:
        - Konstruktor menerima parameter tambahan boolean `hasMSeriesChip` yang menentukan apakah chip seri-M tersedia.
        - Sediakan method `hasMSeriesChip()` bertipe boolean untuk membaca nilai atribut tersebut.
    - **Format Output**:
        - `getAdditionalInfo()` harus mengembalikan string `"Has M-Series Chip: " + (hasMSeriesChip ? "Yes" : "No")`.
        - `printInfo` wajib menampilkan baris `Has M-Series Chip: Yes/No` sesuai nilai atribut sebelum baris `Base Score`.
- **Windows**
    - Enum internal `Edition` dengan nilai `HOME`, `PRO`, `SERVER`.
    - Atribut tambahan: `edition` (Edition). Jika null, gunakan default `HOME`.
    - Aturan multiplier baseScore:
        - Jika `edition == SERVER` dan penggunaan `SERVER` → +30%
        - Untuk penggunaan `GAMING` → +20%
        - Jika `edition == HOME` dan penggunaan `SERVER` → -5%
    - **Format Input**:
        - Konstruktor menerima parameter tambahan `edition`; gunakan `Edition.HOME` apabila nilainya null.
        - Method `getEdition()` harus mengembalikan nilai enum untuk validasi.
    - **Format Output**:
        - `getAdditionalInfo()` harus mengembalikan string `"Edition: " + edition.name()`.
        - `printInfo` wajib menampilkan baris `Edition: <EDITION>` sesuai dengan nilai enum sebelum menampilkan `Base Score`.

Pastikan seluruh skor kompatibilitas akhir dijaga menggunakan `clampScore` agar tidak melebihi 100 atau turun di bawah 0.

Format Output Tambahan

- Method `calculateCompatibility` pada setiap kelas harus mengembalikan nilai akhir bertipe `double` yang sudah diproses oleh `clampScore` sehingga berada di rentang 0–100.

Catatan Tambahan

- Gunakan `println` agar setiap baris output berakhir dengan `\n`.

- Seluruh kelas harus berada di paket default (tanpa deklarasi `package`).
- Gunakan file Main.java untuk melakukan pengujian kelas - kelas yang telah kalian buat.

Upload jawaban sebagai berkas **operating-system.zip** yang berisi keenam file di atas.