1) Namespace + autoload (Composer nélkül, PSR-4 jelleg)

Cél: namespace-ek használata, fájlszerkezet, automatikus osztálybetöltés.

Feladat:

1. Hozzon létre egy src/ mappát, azon belül legalább két namespace-területet, pl.:
   - App\Domain
   - App\Utils
2. Készítsen egy egyszerű autoloader osztályt (pl. Autoloader), ami:
   - regisztrálja a betöltést spl_autoload_register-rel,
   - a namespace-eket fájlútvonalra képezi (PSR-4 logika).
3. Készítsen egy value object jellegű Temperature osztályt:
   - legyenek statikus factory metódusai: Celsiusból / Fahrenheitből,
   - legyenek metódusai visszaalakításra.
4. index.php oldalon:
   - csak az autoloader legyen kézzel require-ölve,
   - a többi osztály automatikusan töltődjön be,
   - űrlapban kérjen egy számot és egy módot (C→F vagy F→C),
   - eredményt táblázatban jelenítse meg.

---

2) Value object-ek + domain objektum: Email, Money, Invoice

Cél: validált, immutábilis értékobjektumok és ezek összekapcsolása domain objektumban.

Feladat:

1. Hozzon létre Email value object-et:
   - konstruktorban validáljon,
   - hibánál dobjon kivételt.
2. Hozzon létre Money value object-et:
   - belül egész Ft értéket tároljon,
   - legyen add(), mul() metódusa,
   - legyen format() metódusa megjelenítéshez.

3. Hozzon létre InvoiceItem (tétel) osztályt:
    - név, egységár (Money), mennyiség,
    - legyen részösszeg számítás.
4. Hozzon létre Invoice osztályt:
    - vevő e-mail (Email),
    - tételek tömbje,
    - nettó és bruttó összeg (pl. 27% ÁFA) számítása.
5. index.php:
    - űrlap: vevő e-mail, tétel név, egységár, mennyiség,
    - beküldés után táblázatban jelenítse meg a tételt és az összesítést,
    - hibákat kezeljen (kivételek + mezőhibák).

---

3) Trait + interface + kivételkezelés egy szolgáltatásban

Cél: trait újrafelhasználás, interface-es DI, kivételdobás/kezelés.

Feladat:

1. Hozzon létre egy TimestampedTrait trait-et:
    - tároljon létrehozási időt,
    - legyen metódus az inicializálásra és kiolvasásra.
2. Hozzon létre LoggerInterface interface-t:
    - info(), error() metódusokkal.
3. Készítsen FileLogger implementációt:
    - logoljon fájlba egységes formátumban.
4. Készítsen Note domain osztályt:
    - használja a trait-et,
    - validáljon (pl. max 200 karakter),
    - hibánál dobjon kivételt.
5. Készítsen NoteService osztályt:
    - konstruktorban kapjon LoggerInterface-t (DI),
    - add() metódus hozza létre a Note-ot és logoljon.

6. index.php:
    - űrlap új jegyzet felvételére,
    - kivételeket kapja el, írjon ki hibát,
    - mutassa az utolsó log sorokat.

---

4) Repository minta + JSON perzisztencia (Student)

Cél: repository interface + JSON implementáció + service réteg + listázás/szűrés/rendezés.

Feladat:

1. Hozzon létre Student entitást:
    - id, név, neptun,
    - neptun validáció (6 karakter A-Z/0-9).
2. Készítsen StudentRepository interface-t:
    - all(), add(), deleteById(), findByNeptun().
3. Készítsen JsonStudentRepository implementációt:
    - fájlba ment JSON-ként,
    - fájlzárat használjon íráskor,
    - objektum ↔ tömb konverzió.
4. Készítsen StudentService réteget:
    - szűrés (keresés név/neptun alapján),
    - rendezés (név/neptun ASC/DESC).
5. index.php:
    - hallgató felvétel (PRG),
    - lista szűréssel és rendezéssel,
    - törlés gomb,
    - teljes adatfájl törlése opció.

---

5) Front controller + router + mini DI + controller/service/repo rétegezés

Cél: „mini keretrendszer” szemlélet: 1 belépési pont, routing, controller, service, repo, view.

Feladat:

1. Készítsen Request osztályt, ami GET/POST/method adatokhoz ad kényelmes hozzáférést.
2. Készítsen Router osztályt:
    - action paraméter alapján route-ol,
    - handler-eket tud regisztrálni.
3. Készítsen egy mini Container (DI) osztályt:
    - service-eket factory-ként regisztrál,
    - egyszer példányosít (singleton jelleg).
4. Készítsen egy egyszerű „Blog” alkalmazást:
    - Post domain osztály (title/body + validáció),
    - PostRepository interface + JSON repo,
    - BlogService üzleti logika,
    - BlogController (list/create/delete).
5. Készítsen views/ sablonokat (layout, list, create).
6. index.php legyen a front controller:
    - container felépítése,
    - router route-ok regisztrálása,
    - dispatch és output.