Həftə 3 — Python OOP: Tapşırıqlar

Gün 1 — Introduction to OOP

Mövzular: Classes, objects, instance methods/attributes, inheritance, polymorphism

Assignment 1.1 — Ticarət Simulyatoru

Bir bazarın ticarət sistemini OOP ilə modelləşdirin.

Initial Data:

```
products_data = [
    ("Buğda", "Ərzaq", 0.85, 10000),
    ("Pambıq", "Xammal", 2.30, 5000),
    ("Neft", "Enerji", 75.50, 200),
    ("Qızıl", "Metal", 1950.00, 50),
    ("Mis", "Metal", 8.20, 3000),
]

traders_data = [
    ("Bakı Ticarət", 50000.00, {"Ərzaq": 0.05, "Xammal": 0.03}),
    ("Gəncə Bazarı", 30000.00, {"Xammal": 0.04, "Metal": 0.06}),
    ("Lənkəran Market", 20000.00, {"Ərzaq": 0.02}),
]
```

Tələblər:

1. `Product` sinfi: `name`, `category`, `base_price`, `stock`, `price_history` (list). Qiymət dəyişdikdə tarixçəyə əlavə olunsun
2. `Trader` sinfi: `name`, `balance`, `inventory` (dict), `discount_rates` (dict — kateqoriyaya görə endirim). Metodlar: `buy(product, qty)`, `sell(product, qty)`, `get_portfolio_value()`
3. `Market` sinfi: `Product` və `Trader` obyektlərini idarə etsin. `execute_trade(trader, product, action, qty)` — alqı-satqı, `update_prices()` — təsadüfi qiymət dəyişikliyi (±10%)
4. **Inheritance**: `PremiumTrader(Trader)` — daha yüksək endirim, kredit limiti. `PerishableProduct(Product)` — istifadə müddəti, keçmişsə satıla bilməz
5. **Polymorphism**: `calculate_fee()` metodu — `Trader` üçün 2%, `PremiumTrader` üçün 1%
6. 10 tur simulyasiya icra edin, hər turda qiymətlər dəyişsin, trader-lər alqı-satqı etsin

Assignment 1.2 — Server Monitoring Sistemi

Bir data mərkəzinin serverlərini izləyən monitoring sistemini modelləşdirin.

Initial Data:

```
servers_config = [
    {"id": "SRV-001", "name": "Web Server", "cpu_cores": 8, "ram_gb": 32, "storage_gb": 500, "role": "web"},
    {"id": "SRV-002", "name": "Database Server", "cpu_cores": 16, "ram_gb": 64, "storage_gb": 2000, "role": "database"},
    {"id": "SRV-003", "name": "Cache Server", "cpu_cores": 4, "ram_gb": 128, "storage_gb": 100, "role": "cache"},
    {"id": "SRV-004", "name": "Worker Node 1", "cpu_cores": 8, "ram_gb": 16, "storage_gb": 250, "role": "worker"},
    {"id": "SRV-005", "name": "Worker Node 2", "cpu_cores": 8, "ram_gb": 16, "storage_gb": 250, "role": "worker"},
]

simulated_metrics = [
    ("SRV-001", {"cpu": 75, "ram": 60, "disk": 45, "network_mbps": 850, "requests_per_sec": 1200}),
    ("SRV-002", {"cpu": 90, "ram": 85, "disk": 70, "network_mbps": 200, "queries_per_sec": 5000}),
    ("SRV-003", {"cpu": 30, "ram": 95, "disk": 15, "network_mbps": 500, "hit_rate": 0.92}),
    ("SRV-004", {"cpu": 95, "ram": 70, "disk": 40, "network_mbps": 100, "jobs_completed": 450}),
    ("SRV-005", {"cpu": 45, "ram": 35, "disk": 38, "network_mbps": 80, "jobs_completed": 380}),
]
```

Tələblər:

1. `Server` sinfi: konfiqurasiya atributları, `status` (online/warning/critical/offline), `metrics_history`, `uptime_hours`. Metodlar: `update_metrics(data)`, `check_health() -> str`, `get_uptime_percentage()`
2. **Inheritance**: `WebServer(Server)` — `requests_per_sec`, `active_connections`. `DatabaseServer(Server)` — `queries_per_sec`, `replication_lag`. `CacheServer(Server)` — `hit_rate`, `eviction_count`
3. **Polymorphism**: `check_health()` hər alt sinifdə fərqli threshold-lar ilə işləsin (WebServer: CPU>80 = warning, DB: RAM>90 = critical, Cache: hit_rate<0.8 = warning)
4. `DataCenter` sinfi: bütün serverləri idarə etsin, `get_overall_health()`, `find_bottlenecks()`, `generate_report()`
5. `Alert` sinfi: xəbərdarlıq yaradın, `severity`, `server_id`, `message`, `timestamp`
6. Metriklər yeniləndikdə avtomatik alert yaratsın

Assignment 1.3 — Bank Hesab Sistemi

Bank hesablarını idarə edən OOP sistemi yazın.

Initial Data:

```
customers_data = [
    {"id": "C-001", "name": "Əli Həsənov", "accounts": [
        {"number": "AZ01-0001", "type": "checking", "balance": 5200.00, "currency": "AZN"},
        {"number": "AZ01-0002", "type": "savings", "balance": 15000.00, "currency": "AZN", "interest_rate": 0.08}
    ]},
    {"id": "C-002", "name": "Leyla Məmmədova", "accounts": [
        {"number": "AZ01-0003", "type": "checking", "balance": 3800.00, "currency": "AZN"},
            {"number": "AZ01-0004", "type": "deposit", "balance": 10000.00, "currency": "USD", "interest_rate": 0.05,
"term_months": 12, "start_date": "2026-01-15"}
    ]},
    {"id": "C-003", "name": "Tural Əliyev", "accounts": [
        {"number": "AZ01-0005", "type": "savings", "balance": 800.00, "currency": "AZN", "interest_rate": 0.06}
    ]},
]
```

Tələblər:

1. `Account` sinfi: `number`, `balance`, `currency`, `transaction_history` (list of tuples). Metodlar: `deposit(amount)`, `withdraw(amount)`, `get_statement()`
2. **Inheritance**: `CheckingAccount(Account)` — overdraft limiti, aylıq xidmət haqqı. `SavingsAccount(Account)` — faiz dərəcəsi, minimum balans tələbi, `calculate_interest()`. `DepositAccount(Account)` — müddət, vaxtından əvvəl çıxarışda cərimə
3. **Polymorphism**: `apply_monthly_rules()` — Checking: xidmət haqqı tutulur, Savings: faiz hesablanır, Deposit: müddət yoxlanır
4. `Customer` sinfi: müştəri hesablarını idarə etsin, `get_total_balance()`, `transfer(from_acc, to_acc, amount)`
5. `Bank` sinfi: bütün müştəriləri idarə etsin, `find_customer(name)`, `get_total_deposits()`, `end_of_month_processing()` — bütün hesablar üçün aylıq qaydaları tətbiq etsin
6. Hər əməliyyatı `transaction_history`-ə yazın: `(date, type, amount, balance_after, description)`

Bonus Assignment 1.B — Refaktor: Advanced OOP ilə Genişləndirmə

> *Gün 2-nin mövzularını öyrəndikdən sonra edin.*

Assignment 1.2 (Server Monitoring) kodunu refaktor edin:

1. `Server` sinfinə `@property` əlavə edin: `cpu_usage`, `ram_usage` — setter ilə validasiya (0-100)
2. `@classmethod from_config(cls, config_dict)` — dict-dən server yaratma
3. `@staticmethod format_bytes(value)` — baytı oxunaqlı formata çevirmə
4. Decorator ilə sinifə funksionallıq əlavə edin: `@log_method_calls` — hər metod çağırışını log-lasın
5. **Composition**: `Server` sinfinə `CPUMonitor`, `RAMMonitor`, `DiskMonitor` komponent obyektləri əlavə edin (inheritance əvəzinə)

Gün 2 — Advanced OOP Concepts

Mövzular: class methods, static methods, properties, decorators with classes, inheritance vs composition

---

Assignment 2.1 — Plugin Sistemi

Genişləndirilə bilən plugin arxitekturası yazın.

Tələblər:

1. `Plugin` base sinfi: `name`, `version`, `enabled`, `@property priority`, `@abstractmethod execute(data)`, `@classmethod get_info(cls) -> str`
2. **Concrete Plugin-lər:**
   - `TextTransformPlugin` — mətn əməliyyatları (uppercase, reverse, word count). Composition: `TextProcessor` komponentindən istifadə etsin

- `MathPlugin` — riyazi hesablamalar (factorial, prime check, stats). `@staticmethod` ilə utility funksiyalar
- `ValidationPlugin` — data validasiyası (email, phone, date format). `@property` ilə son validasiya nəticəsi

1. `PluginManager` sinfi:
   - `register(plugin)`, `unregister(name)` — plugin idarəsi
   - `@classmethod from_config(cls, config_dict)` — konfiqurasiyadan yüklə
   - `execute_pipeline(data, plugin_names)` — plugin-ləri ardıcıl icra etsin
   - `get_stats()` — hər plugin-in neçə dəfə icra edildiyini göstərsin
1. **Decorators with classes**: `@plugin_logger` — hər plugin icrasını log etsin, `@plugin_timer` — vaxtını ölçsün
2. `PluginConfig` sinfi: JSON fayldan plugin konfiqurasiyası oxusun, `@property` ilə validasiya

Initial Data:

```
pipeline_config = {
    "plugins": [
        {"name": "TextTransform", "version": "1.0", "enabled": True, "priority": 1, "settings": {"operation": "uppercase"}},
            {"name": "Validation", "version": "1.2", "enabled": True, "priority": 2, "settings": {"rules": ["not_empty",
"max_length:100"]}},
        {"name": "Math", "version": "2.0", "enabled": False, "priority": 3, "settings": {"precision": 4}},
    ],
    "pipeline_order": ["Validation", "TextTransform"]
}

test_data = [
    {"text": "Hello World", "numbers": [1, 2, 3, 4, 5], "email": "test@example.com"},
    {"text": "", "numbers": [], "email": "invalid-email"},
    {"text": "Python OOP", "numbers": [10, 20, 30], "email": "user@academy.az"},
]
```

Assignment 2.2 — Şablon Mühərriki (Template Engine)

Sadə şablon mühərriki yazın — HTML/text şablonlarını data ilə dolduran sistem.

Tələblər:

1. `Template` sinfi:
   - `__init__(self, template_str)` — şablon mətnini qəbul etsin
   - `render(self, context: dict) -> str` — `{{variable}}` ifadələrini context-dən əvəz etsin
   - `@property variables` — şablondakı bütün dəyişənləri set olaraq qaytarsın
   - `@classmethod from_file(cls, filepath)` — fayldan şablon oxusun
   - `@staticmethod escape_html(text)` — HTML simvollarını escape etsin

1. **Inheritance:**
   - `ConditionalTemplate(Template)` — `{% if condition %}...{% endif %}` dəstəkləsin
   - `LoopTemplate(Template)` — `{% for item in items %}...{% endfor %}` dəstəkləsin
   - `FullTemplate(ConditionalTemplate, LoopTemplate)` — hər ikisini birləşdirsin (multiple inheritance)

1. **Composition:**
   - `TemplateCache` sinfi — render edilmiş nəticələri cache-ləsin
   - `TemplateLoader` sinfi — fayldan şablonları yükləsin, `os.path` ilə
   - `TemplateEngine` sinfi — `TemplateCache` + `TemplateLoader` birləşdirsin

1. `@property is_valid` — şablonun bütün tag-ları düzgün bağlanıb-bağlanmadığını yoxlasın

2. Context-də olmayan dəyişən üçün `TemplateSyntaxError` raise edin

Initial Data:

```
templates = {
    "report": """
    <h1>{{title}}</h1>
    <p>Tarix: {{date}}</p>
    {% if show_summary %}
    <div class="summary">
        <p>Ümumi: {{total}} element</p>
        <p>Status: {{status}}</p>
    </div>
    {% endif %}
    <ul>
    {% for item in items %}
        <li>{{item.name}} - {{item.value}} AZN</li>
    {% endfor %}
    </ul>
    """,
    "email": """
    Hörmətli {{name}},
    {{message}}
    {% if has_attachment %}
    Əlavə: {{attachment_name}}
    {% endif %}
    Hörmətlə, {{sender}}
    """,
}

contexts = {
    "report": {
        "title": "Aylıq Hesabat",
        "date": "2026-03-17",
        "show_summary": True,
        "total": 15,
        "status": "Tamamlandı",
        "items": [
            {"name": "Server xərcləri", "value": 450},
            {"name": "Lisenziya", "value": 200},
            {"name": "Əmək haqqı", "value": 8500},
        ]
    },
    "email": {
        "name": "Əli Həsənov",
        "message": "Layihə uğurla tamamlandı.",
        "has_attachment": True,
        "attachment_name": "report.pdf",
        "sender": "Admin"
    }
}
```

Assignment 2.3 — Sensor Şəbəkəsi Simulyatoru

IoT sensor şəbəkəsini OOP ilə modelləşdirin. Composition prioritet olsun.

Tələblər:

1. **Komponent sinifləri** (Composition üçün):
   - `Battery`: `level`, `drain_rate`, `charge()`, `@property is_low`
   - `Transmitter`: `frequency`, `range_m`, `signal_strength`, `send(data)`, `@staticmethod calculate_signal_loss(distance)`
   - `DataLogger`: `storage_capacity`, `records`, `log(record)`, `export(format)`, `@classmethod from_existing(cls, records)`
1. `Sensor` sinfi — `Battery`, `Transmitter`, `DataLogger` komponentlərini saxlasın:
   - `@property status` — batareya, siqnal, yaddaşa görə ümumi status
   - `read()` → abstract
   - `@classmethod create_network(cls, configs)` — bir çox sensor yaratma
   - `calibrate()`, `reset()`
1. **Inheritance (sensor növləri):**
   - `TemperatureSensor(Sensor)` — dərəcə, `@property celsius/fahrenheit` (çevirici property)
   - `HumiditySensor(Sensor)` — rütubət, `@staticmethod dew_point(temp, humidity)`
   - `MotionSensor(Sensor)` — hərəkət, cooldown müddəti, `@property is_triggered`
1. `SensorNetwork` sinfi — bütün sensorları idarə etsin:
   - `add_sensor(sensor)`, `remove_sensor(id)`
   - `collect_all_readings()` — bütün sensorlardan data toplasın
   - `get_alerts()` — aşağı batareya, zəif siqnal, dolu yaddaş xəbərdarlıqları
   - `generate_heatmap_data()` — temperatur sensorlarının datası
1. Dekorator: `@requires_power` — batareya bitibsə əməliyyatı bloklasın, `@log_reading` — hər oxunuşu log-lasın

Initial Data:

```
sensor_configs = [
    {"id": "S-001", "type": "temperature", "location": "Otaq A", "battery": 85, "range": 50},
    {"id": "S-002", "type": "humidity", "location": "Otaq A", "battery": 62, "range": 50},
    {"id": "S-003", "type": "temperature", "location": "Otaq B", "battery": 15, "range": 30},
    {"id": "S-004", "type": "motion", "location": "Dəhliz", "battery": 93, "range": 20},
    {"id": "S-005", "type": "temperature", "location": "Server Otağı", "battery": 45, "range": 50},
    {"id": "S-006", "type": "humidity", "location": "Anbar", "battery": 8, "range": 40},
    {"id": "S-007", "type": "motion", "location": "Giriş", "battery": 77, "range": 25},
]
```

Bonus Assignment 2.B — Refaktor: Concurrency ilə Paralel Oxuma

> **Gün 3-ün mövzularını öyrəndikdən sonra edin.**

Assignment 2.3 (Sensor Şəbəkəsi) kodunu refaktor edin:

1. `threading.Thread` ilə hər sensoru paralel oxuyun
2. `threading.Lock` ilə paylaşılan `DataLogger`-ə yazarkən race condition qarşısını alın
3. `concurrent.futures.ThreadPoolExecutor` ilə sensor pooling tətbiq edin
4. Sensor oxunuşlarını `queue.Queue` ilə ardıcıl emal edin
5. Performans müqayisəsi: ardıcıl vs paralel oxuma vaxtını ölçün

Gün 3 — Concurrency (Threading & Multiprocessing)

Mövzular: threading module, multiprocessing module, thread/process management

Assignment 3.1 — Paralel Fayl Emal Sistemi

Böyük fayl dəstlərini paralel emal edən sistem yazın.

Initial Data — proqram bu faylları avtomatik yaratsın:

```
import random, string
def generate_test_files(directory, count=20):
    """Test faylları yarat: hər birində təsadüfi mətn, ədədlər, email ünvanları"""
    domains = ["gmail.com", "yahoo.com", "academy.az", "mail.ru", "outlook.com"]
    words = ["python", "data", "server", "network", "module", "thread", "process",
             "function", "class", "object", "variable", "loop", "condition", "error"]
    for i in range(count):
        lines = []
        for _ in range(random.randint(100, 500)):
            line_type = random.choice(["text", "number", "email", "mixed"])
            if line_type == "text":
                lines.append(" ".join(random.choices(words, k=random.randint(5, 15))))
            elif line_type == "number":
                lines.append(" ".join(str(random.randint(1, 10000)) for _ in range(random.randint(3, 10))))
            elif line_type == "email":
                name = "".join(random.choices(string.ascii_lowercase, k=8))
                lines.append(f"Contact: {name}@{random.choice(domains)} | ID: {random.randint(1000,9999)}")
            else:
                lines.append(f"{random.choice(words)}: {random.randint(1,100)} - {''.join(random.choices(string.ascii_letters,
k=20))}")
        filepath = os.path.join(directory, f"data_{i:03d}.txt")
        with open(filepath, "w", encoding="utf-8") as f:
            f.write("\n".join(lines))
```

Tələblər:

1. **Ardıcıl emal** (`sequential_process`): hər faylı bir-bir oxuyun, hər faylda: söz sayı, sətir sayı, email ünvanları, ən çox istifadə olunan söz, ədədlərin cəmi
2. **Threading ilə** (`threaded_process`): `threading.Thread` ilə hər fayl üçün ayrı thread yaradın. `threading.Lock` ilə paylaşılan nəticə dict-ə yazın
3. **ThreadPoolExecutor ilə** (`pool_process`): `concurrent.futures.ThreadPoolExecutor(max_workers=5)` istifadə edin
4. **Multiprocessing ilə** (`multiprocess_process`): `multiprocessing.Process` ilə icra edin, nəticələri `multiprocessing.Queue` ilə toplayın
5. **ProcessPoolExecutor ilə**: `concurrent.futures.ProcessPoolExecutor` ilə
6. Hər metodun icra vaxtını ölçün, müqayisə cədvəli çıxarın:

```
Metod                Vaxt (san)  Sürət artımı

Ardıcıl                 4.52     1.00x
Threading               2.10     2.15x
ThreadPool(5)           1.85     2.44x
Multiprocessing         1.20     3.77x
ProcessPool(5)          1.05     4.30x
```

Assignment 3.2 — Real-time Məlumat Toplayıcısı

Bir neçə "data source"-dan eyni vaxtda məlumat toplayan real-time sistem yazın. Data source-lar simulyasiya edilir.

Tələblər:

1. `DataSource` sinfi: hər source fərqli tezliklə data generasiya etsin (simulyasiya):

```
sources_config = [
            {"id":   "stock",   "name":   "Birja   Qiymətləri",   "interval":   0.5,   "data_gen":   lambda:   {"symbol":
random.choice(["AAPL","GOOGL","MSFT"]), "price": round(random.uniform(100,500),2)}},
    {"id": "weather", "name": "Hava Məlumatı", "interval": 2.0, "data_gen": lambda: {"city": random.choice(["Bakı","Gəncə"]),
"temp": round(random.uniform(5,35),1)}},
            {"id":   "traffic",   "name":   "Trafik   Sensorları",   "interval":   1.0,   "data_gen":   lambda:   {"road":
random.choice(["M1","M2","M3"]), "vehicles": random.randint(10,200)}},
            {"id":   "social",   "name":   "Sosial   Media",   "interval":   0.3,   "data_gen":   lambda:   {"platform":
random.choice(["Twitter","Reddit"]), "mentions": random.randint(0,500)}},
]
```

1. Hər data source üçün ayrı **thread** yaradın, `threading.Event` ilə dayandırma
2. `queue.Queue` ilə bütün source-lardan gələn datanı bir yerə toplayın (producer-consumer pattern)
3. **Consumer thread** — queue-dan datanı oxuyub emal etsin, aqreqasiya etsin
4. `threading.Timer` ilə hər 5 saniyədən bir aralıq hesabat çap edin
5. `threading.Barrier` ilə bütün source-ların hazır olmasını gözləyin, sinxron başlasınlar
6. 15 saniyə sonra `Event.set()` ilə bütün thread-ləri dayandırın, final hesabat çıxarın

---

Assignment 3.3 — Şəkil Emal Pipeline (CPU-intensive)

CPU-yüklü əməliyyatları `multiprocessing` ilə paralel icra edən sistem yazın. Şəkil əvəzinə rəqəmsal matrislərlə işləyin.

Tələblər:

1. "Şəkil" = 2D list (matris). Generasiya:

```
def generate_image(width, height):
    return [[random.randint(0, 255) for _ in range(width)] for _ in range(height)]

images = [generate_image(200, 200) for _ in range(10)]
```

1. Emal funksiyaları (CPU-intensive):

- `blur(image, radius)` — orta dəyər filtri (hər piksel = ətrafındakıların ortası)
- `sharpen(image)` — kəskinləşdirmə (convolution kernel ilə)
- `edge_detect(image)` — kənar algılama (Sobel operator, sadələşdirilmiş)
- `histogram(image) -> dict` — piksel dəyərlərinin tezliyi
- `brightness(image, factor)` — parlaqlıq dəyişdirmə

1. **Ardıcıl emal**: bütün şəkilləri bir-bir emal edin
2. **multiprocessing.Pool** ilə: `pool.map(process_image, images)` — paralel emal
3. **multiprocessing.Process + Pipe** ilə: ayrı proseslər arası nəticə ötürmə
4. `multiprocessing.shared_memory` (və ya `Manager`) ilə paylaşılan statistika
5. CPU core sayına görə optimal pool size hesablayın: `os.cpu_count()`
6. Performans müqayisəsi: ardıcıl vs Pool(2) vs Pool(4) vs Pool(CPU_count)

Həftəlik Assignment — Quest of Shadows: AI NPC Macəra Oyunu

> *Bütün həftənin mövzularını əhatə edir: OOP (inheritance, polymorphism, composition, properties, class/static methods, decorators), və Concurrency (threading).*

Oyunun Təsviri

Terminal əsaslı quest oyunu yazın. Oyunçu qəhrəman yaradır, quest-lər alır, döyüşlər edir, NPC-lərlə OpenRouter API vasitəsilə söhbət edir.

Paket strukturu:

```
quest_of_shadows/
 __init__.py
 entities/
    __init__.py
    character.py    # Qəhrəman və düşmən sinifləri
    npc.py          # NPC sinifləri + AI
    items.py        # Əşya sinifləri
 world/
    __init__.py
    locations.py    # Məkan sinifləri
    quests.py       # Quest sistemi
    events.py       # Hadisə sistemi
 systems/
    __init__.py
    combat.py       # Döyüş sistemi
    inventory.py    # İnventar sistemi
    save_load.py    # Saxlama/yükləmə (File I/O)
 ai/
    __init__.py
    npc_brain.py    # OpenRouter API ilə NPC zekası
 concurrency/
    __init__.py
    background.py   # Thread ilə fon prosesləri
 config.py
 main.py
```

Initial Data

```
# ===== MƏKANLAR =====
locations_data = {
    "village": {
        "name": "Kölgələr Kəndi",
        "description": "Dağ ətəyində sakit bir kənd. Bazarda tacirlər var, mərkəzdə bulaq axır.",
        "connected": ["forest", "marketplace"],
        "danger_level": 0,
    },
    "marketplace": {
        "name": "Böyük Bazar",
        "description": "Cürbəcür əşyalar satan tacirlər burada toplaşıb.",
        "connected": ["village"],
        "danger_level": 0,
    },
    "forest": {
        "name": "Qaranlıq Meşə",
        "description": "Sıx ağaclar arasında işıq zor keçir. Heyvan səsləri eşidilir.",
        "connected": ["village", "cave", "river"],
        "danger_level": 3,
    },
    "cave": {
        "name": "Əjdaha Mağarası",
        "description": "Qayalıqlar arasında gizlənmiş nəhəng mağara. İçəridən istilik gəlir.",
        "connected": ["forest"],
        "danger_level": 7,
    },
    "river": {
        "name": "Gümüş Çay",
        "description": "Geniş çay sahili. Balıqçılar burada toplaşır.",
        "connected": ["forest", "tower"],
        "danger_level": 2,
    },
    "tower": {
        "name": "Qədim Qüllə",
        "description": "Xarabalığa çevrilmiş qüllə. Ən yuxarıda sirli işıq var.",
        "connected": ["river"],
        "danger_level": 9,
    },
}

# ===== NPC-LƏR =====
npcs_data = {
    "elder": {
        "name": "Ağsaqqal Mirəli",
        "role": "Quest verən",
        "location": "village",
        "personality": "Müdrik, az danışan, sirli. Kəndin tarixini yaxşı bilir. Hər sözün arxasında dərin məna var.",
          "knowledge": "Əjdaha mağarasındakı xəzinə haqqında bilir. Qədim qüllənin sirrini bilir. Kölgələr Kəndinin keçmişi
haqqında danışa bilər.",
    },
    "merchant": {
        "name": "Tacir Rüfət",
        "role": "Alver edən",
        "location": "marketplace",
          "personality": "Zarafatcıl, ticarətə həris, amma insaflı. Bəzən maraqlı söhbətlər edir. Əşyalar haqqında detallı
məlumat verir.",
        "knowledge": "Bazardakı bütün əşyaların keyfiyyətini bilir. Meşədə nadir bitkilər olduğunu eşidib. Qüllədən gələn işıq
haqqında şayiə yayır.",
    },
    "hermit": {
```

```
        "name": "Tənha Könül",
        "role": "Müdrik",
        "location": "river",
        "personality": "Sakit, filosofik, təbiətsevər. Metaforlarla danışır. Bəzən cavab əvəzinə sual verir.",
          "knowledge": "Meşədəki heyvanları yaxşı tanıyır. Çayın şəfa verən xüsusiyyətlərini bilir. Qüllədəki işığın qaynağı haqqında fikirlər irəli sürür.",
    },
    "guard": {
        "name": "Keşikçi Anar",
        "role": "Mühafizəçi",
        "location": "forest",
        "personality": "Cəsur, sadiq, az sözlü. Kəndi qorumaq üçün hər şey edər. Döyüş haqqında məsləhət verə bilir.",
            "knowledge": "Meşədəki düşmənlərin yerini bilir. Mağaraya gedən təhlükəsiz yol haqqında məlumatı var. Döyüş taktikalarını öyrədə bilər.",
    },
}

# ===== ƏŞYALAR =====
items_data = {
    "iron_sword": {"name": "Dəmir Qılınc", "type": "weapon", "damage": 15, "price": 120, "durability": 100},
    "steel_shield": {"name": "Polad Qalxan", "type": "armor", "defense": 10, "price": 90, "durability": 80},
    "health_potion": {"name": "Şəfa İksiri", "type": "consumable", "heal": 50, "price": 30, "quantity": 1},
    "mana_potion": {"name": "Mana İksiri", "type": "consumable", "mana_restore": 30, "price": 25, "quantity": 1},
    "fire_scroll": {"name": "Od Tilsimi", "type": "consumable", "damage": 40, "price": 75, "quantity": 1},
    "ancient_map": {"name": "Qədim Xəritə", "type": "quest", "description": "Qüllənin gizli girişini göstərir", "price": 200},
     "dragon_scale": {"name": "Əjdaha Pulcuğu", "type": "material", "description": "Çox nadir, güclü zireh üçün material", "price": 500},
    "crystal_amulet": {"name": "Kristal Talisman", "type": "accessory", "effect": "mana_regen", "bonus": 5, "price": 350},
}

# ===== DÜŞMƏNLƏR =====
enemies_data = {
    "wolf": {"name": "Vəhşi Canavar", "hp": 40, "damage": 8, "defense": 3, "xp": 20, "loot": ["health_potion"]},
    "bandit": {"name": "Yol Kəsən", "hp": 60, "damage": 12, "defense": 5, "xp": 35, "loot": ["health_potion", "iron_sword"]},
    "spider": {"name": "Nəhəng Hörümçək", "hp": 50, "damage": 15, "defense": 2, "xp": 30, "loot": ["mana_potion"]},
        "golem": {"name": "Daş Qolem", "hp": 120, "damage": 20, "defense": 15, "xp": 80, "loot": ["steel_shield", "crystal_amulet"]},
       "dragon": {"name": "Kölgə Əjdahası", "hp": 300, "damage": 35, "defense": 20, "xp": 500, "loot": ["dragon_scale", "fire_scroll", "ancient_map"]},
}

# ===== QUESTLƏR =====
quests_data = [
    {
        "id": "Q-001", "title": "Meşənin Təhlükəsi",
        "giver": "elder", "description": "Meşədən 3 canavar öldür və kəndə qayıt.",
        "objectives": [{"type": "kill", "target": "wolf", "count": 3}],
        "reward": {"xp": 100, "gold": 150, "items": ["health_potion"]},
        "prerequisite": None
    },
    {
        "id": "Q-002", "title": "Tacirin Sifarişi",
        "giver": "merchant", "description": "Meşədən nadir bitki tap və Tacir Rüfətə gətir.",
        "objectives": [{"type": "collect", "target": "rare_herb", "count": 5}],
        "reward": {"xp": 80, "gold": 200, "items": ["mana_potion"]},
        "prerequisite": None
    },
    {
        "id": "Q-003", "title": "Mağara Sirri",
        "giver": "elder", "description": "Mağaraya gir, Daş Qolemi məğlub et, qədim artefaktı tap.",
          "objectives": [{"type": "kill", "target": "golem", "count": 1}, {"type": "collect", "target": "ancient_artifact",
```

```
    "count": 1}],
            "reward": {"xp": 250, "gold": 400, "items": ["crystal_amulet"]},
            "prerequisite": "Q-001"
        },
        {
            "id": "Q-004", "title": "Kölgə Əjdahası",
            "giver": "hermit", "description": "Qüllənin zirvəsinə qalx və Kölgə Əjdahasını məğlub et.",
            "objectives": [{"type": "kill", "target": "dragon", "count": 1}],
            "reward": {"xp": 1000, "gold": 2000, "items": ["dragon_scale"]},
            "prerequisite": "Q-003"
        },
    ]
```

Tələblər

Hissə 1 — OOP Əsasları (Inheritance, Polymorphism)

1. `Character` base sinfi: `name`, `hp`, `max_hp`, `level`, `xp`, `gold`, `inventory`. Metodlar: `take_damage(amount)`, `heal(amount)`, `is_alive()`, `level_up()`
2. `Hero(Character)`: əlavə `mana`, `equipped_weapon`, `equipped_armor`, `quests`. `attack()` polymorphism — silaha görə dəyişir
3. `Enemy(Character)`: `loot_table`, `danger_level`, `respawn()`. Fərqli düşmən tipləri `attack()` metodunu override etsin
4. `Item` base sinfi → `Weapon(Item)`, `Armor(Item)`, `Consumable(Item)`, `QuestItem(Item)` — hər biri `use()` metodunu polymorphic implement etsin

Hissə 2 — Advanced OOP (Properties, Class/Static Methods, Composition, Decorators)

1. `@property hp` — setter ilə 0-dan aşağı düşməsin, `max_hp`-dan yuxarı qalxmasın
2. `@classmethod Hero.create_default(cls, name)` — standart stat-larla qəhrəman yaratma
3. `@staticmethod Character.calculate_damage(attacker_dmg, defender_def)` — zərər formulası
4. **Composition**: `Inventory` sinfi (Character-ə daxil), `QuestLog` sinfi (Hero-ya daxil), `LootTable` sinfi (Enemy-yə daxil)
5. **Decorators**: `@requires_alive` — ölü character əməliyyat edə bilməz, `@costs_mana(amount)` — mana yetərsizdirsə blokla, `@log_combat` — döyüş hadisələrini log-la

Hissə 3 — AI NPC Sistemi (OpenRouter API)

1. `NPCBrain` sinfi — OpenRouter API ilə NPC danışığını idarə etsin:

```
class NPCBrain:
    def __init__(self, api_key, model="google/gemma-3-4b-it:free"):
        self.api_key = api_key
        self.conversation_history = []

    def generate_response(self, npc_data, player_message, game_context):
        system_prompt = f"""Sən "{npc_data['name']}" adlı NPC-sən.
        Rolun: {npc_data['role']}
        Şəxsiyyətin: {npc_data['personality']}
        Bildiyin məlumatlar: {npc_data['knowledge']}

        Oyunçunun hazırkı vəziyyəti:
        - Səviyyə: {game_context['level']}, HP: {game_context['hp']}/{game_context['max_hp']}
        - Məkan: {game_context['location']}
        - Aktiv questlər: {game_context['active_quests']}

        QAYDALAR:
        - Həmişə öz rolunda qal, xarakterindən çıxma
        - Qısa və oyun dünyasına uyğun cavab ver (2-4 cümlə)
        - Lazım gəldikdə quest və ya ipucu ver
        - Azərbaycan dilində cavab ver"""
        # API call...
```

1. Hər NPC-nin `conversation_history` olsun — əvvəlki söhbəti xatırlasın
2. API çağırışı uğursuz olarsa **fallback** cavablar istifadə edin (offline rejim)
3. NPC ilə söhbət: `talk_to_npc(npc_id)` — interaktiv söhbət loop-u

Hissə 4 — Concurrency (Threading)

1. **Fon prosesləri** (ayrı thread-lərdə):
    - `auto_save_thread` — hər 60 saniyədə oyunu avtomatik saxlasın (`threading.Timer`)
    - `event_thread` — təsadüfi hadisələr generasiya etsin (hər 30 saniyədə: hava dəyişikliyi, NPC hərəkəti, düşmən spawn)
    - `regen_thread` — qəhrəmanın HP/Mana-sını yavaş-yavaş bərpa etsin (hər 10 saniyədə)
1. `threading.Event` ilə thread-ləri dayandırma (oyun bitdikdə)
2. `threading.Lock` ilə qəhrəman stat-larına paralel daxilolmanı qoruma
3. `queue.Queue` ilə hadisə bildirişlərini əsas oyun loop-una ötürmə

Hissə 5 — Save/Load (File I/O)

1. `save_game(hero, world_state)` — oyun vəziyyətini JSON faylına yazın
2. `load_game(filepath) -> (hero, world_state)` — fayldan yükləyin
3. Custom exception-lar: `SaveCorruptedError`, `IncompatibleVersionError`

Oyun Loop Nümunəsi

```
      QUEST OF SHADOWS — Kölgələr Macərası


 Kölgələr Kəndi
   Dağ ətəyində sakit bir kənd. Bazarda tacirlər
   var, mərkəzdə bulaq axır.

 Kənan (Səviyyə 3) |  85/100 |  12/30 |  250

 Keçidlər: [1] Qaranlıq Meşə  [2] Böyük Bazar
 NPC-lər: [A] Ağsaqqal Mirəli
 Aktiv Quest: Meşənin Təhlükəsi (1/3 canavar)

> Nə etmək istəyirsiniz?
  [1-2] Getmək  [A] Danışmaq  [I] İnventar
  [Q] Questlər  [S] Saxla     [X] Çıxış

> A

   Ağsaqqal Mirəli ilə söhbət

Mirəli: "Xoş gəlmisən, gənc döyüşçü. Meşədəki
canavarlar kəndi narahat edir..."

> Mağara haqqında nə bilirsən?

Mirəli: "Mağara... Orada qədim vaxtlardan qalan
bir şey var. Amma əvvəlcə özünü sübut etməlisən.
Meşədəki canavarları məğlub et, sonra danışarıq."

> çıx


 [Hadisə] Hava dəyişdi: Yağış başladı
 [Bərpa] HP: 85 → 90

>
```