

ЭМНЭЛГИЙН БҮРТГЭЛИЙН СИСТЕМ

Hospital Registration System



Танилцуулга

Энэ төсөл нь эмнэлгийн оочир дараалал, бүртгэлийг бүрэн хөнгөлөхөд тулгуурласан. Системийн хувьд өвчтөн бүртгэх, эмч дээрх сул цагийг харах, эмч өвчтөний түүх болон эмчилгээ, хэрэглэх эм бэлдмэлийн дэлгэрэнгүй мэдээллийг бичиж хадгалах үйл явцыг хялбаршуулсан.

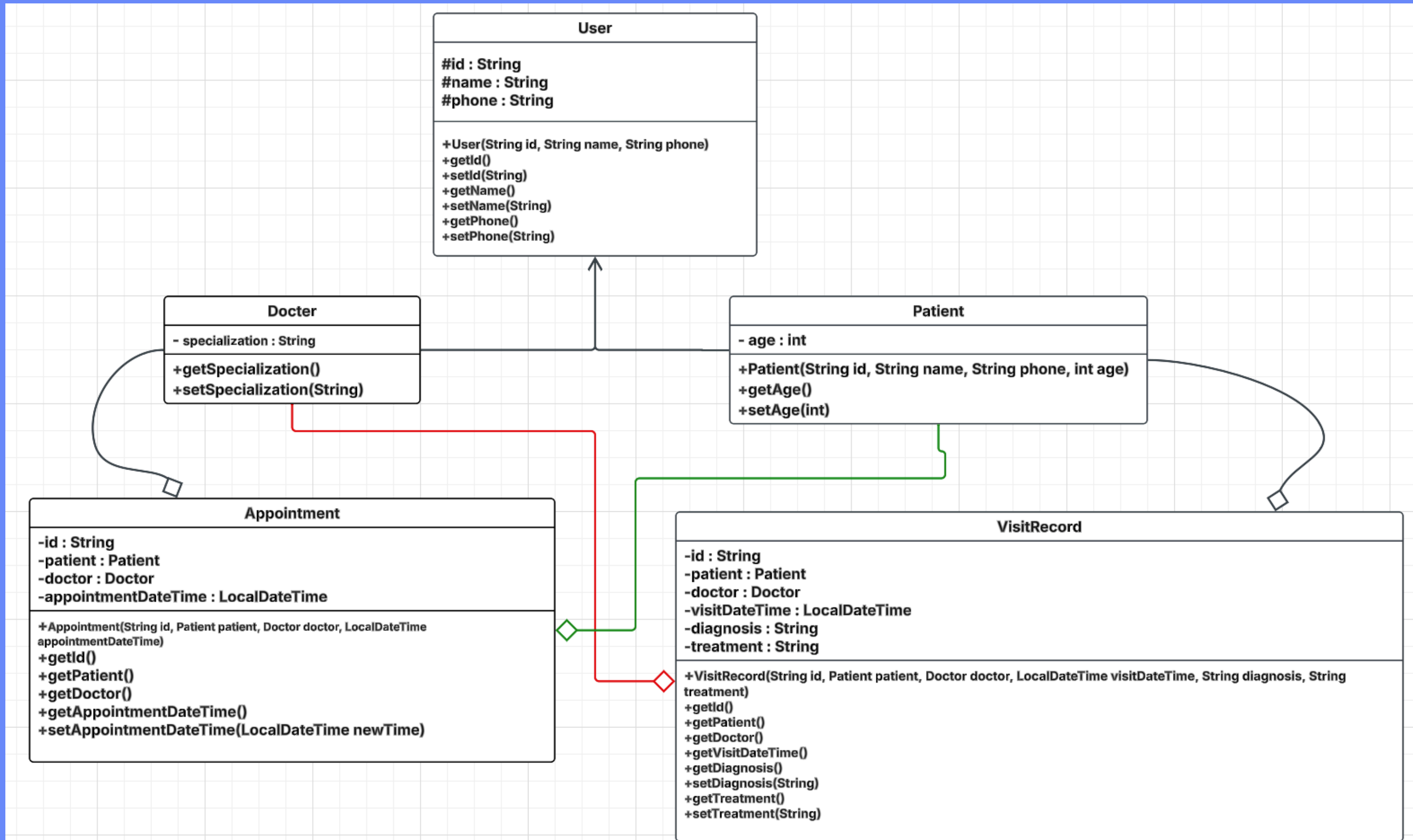




Гол онцлогууд:

- Өвчтөн нэмэх, засах, устгах
- Эмчийн мэдээлэл бүртгэх
- Цаг захиалгын систем (Appointment)
- Өвчтөний түүх хадгалах (VisitRecord)
- Мэдээлэл хайх, жагсаалтаар харах
- Ирц, онош, эмчилгээний түүхийн менежмент

UML DIAGRAM



USER.JAVA

Энэ класс нь эцэг класс бөгөөд эмч, өвчтөн хоёр дээрх кодны давхцалыг арилгаж байгуулагч функц болон “get”, “set” функцуудыг агуулна.

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

abstract class User {
    private static final Logger logger = LogManager.getLogger(clazz:User.class);

    protected String id;
    protected String name;
    protected String phone;

    public User(String id, String name, String phone) {
        if (id == null || id.trim().isEmpty()) {
            logger.error(message:"User creation failed: empty id");
            throw new IllegalArgumentException(s:"ID cannot be empty.");
        }
        this.id = id;
        this.name = name;
        this.phone = phone;
        logger.info(message:"User created: id={}, name={}", id, name);
    }
}
```

DOCTOR.JAVA

Энэ класс нь USER
класссаас
удамшиж эмчийн
мэргэшсэн
чиглэлийг
авах функцууд
агуулсан.

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class Doctor extends User {
    private static final Logger logger = LogManager.getLogger(clazz:Doctor.class);

    private String specialization;

    public Doctor(String id, String name, String phone, String specialization) {
        super(id, name, phone);
        setSpecialization(specialization);
        logger.info(message:"Doctor created: id={}, name={}, specialization={}", id, name, specialization);
    }

    public String getSpecialization() {
        logger.debug(message:"getSpecialization called: {}", specialization);
        return specialization;
    }

    public void setSpecialization(String specialization) {
        if (specialization == null || specialization.trim().isEmpty()) {
            logger.error(message:"Invalid specialization: empty");
            throw new IllegalArgumentException(s:"The specialization field cannot be empty.");
        }
        this.specialization = specialization;
        logger.info(message:"Doctor specialization updated: {}", specialization);
    }
}
```

PATIENT.JAVA

Энэ класс нь USER
класссаас
удамшиж ӨВЧТНИЙ
мэдээлэл авах болон
нас авах "get", "set"
функц бичигдсэн.

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class Patient extends User {
    private static final Logger logger = LogManager.getLogger(clazz:Patient.class);

    private int age;

    public Patient(String id, String name, String phone, int age) {
        super(id, name, phone);
        setAge(age);
        logger.info(message:"Patient created: id={}, name={}", id, name);
    }

    public int getAge() {
        logger.debug(message:"getAge called: {}", age);
        return age;
    }

    public void setAge(int age) {
        if (age < 0) {
            logger.error(message:"Invalid age: {}", age);
            throw new IllegalArgumentException(s:"Age cannot be negative.");
        }
        this.age = age;
        logger.info(message:"Patient age updated: {}", age);
    }
}
```

APPOINTMENT.JAVA

Энэ класс нь захиалгын
дугаар, өвчтөн, сонгосон
эмч, он сар өдрийг агуулж
эдгээр функцийн
"get", "set" бичигдсэн.

```
import java.time.LocalDateTime;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class Appointment {

    private static final Logger logger = LogManager.getLogger(clazz:Appointment.class);

    private String id;
    private Patient patient;
    private Doctor doctor;
    private LocalDateTime appointmentDateTime;

    public Appointment(String id, Patient patient, Doctor doctor, LocalDateTime appointmentDateTime) {
        if (id == null || id.trim().isEmpty()) {
            throw new IllegalArgumentException(s:"Appointment ID cannot be null or empty");
        }
        if (patient == null) {
            throw new IllegalArgumentException(s:"Patient cannot be null");
        }
        if (doctor == null) {
            throw new IllegalArgumentException(s:"Doctor cannot be null");
        }
        if (appointmentDateTime == null) {
            throw new IllegalArgumentException(s:"Appointment date/time cannot be null");
        }

        this.id = id;
        this.patient = patient;
        this.doctor = doctor;
        this.appointmentDateTime = appointmentDateTime;

        logger.info("Appointment created: " + id);
    }
}
```


VisitRecord.JAVA

Энэ класс нь үзлэгийн,
дугаар, үзүүлсэн өвчтөн,
үзлэг хийсэн эмч,
үзлэгийн огноо,
онош, эмчилгээний
мэдээлэл хадгалах
функц

```
import java.time.LocalDateTime;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class VisitRecord {

    private static final Logger logger = LogManager.getLogger(clazz:VisitRecord.class);

    private String id;
    private Patient patient;
    private Doctor doctor;
    private LocalDateTime visitDateTime;
    private String diagnosis;
    private String treatment;

    public VisitRecord(String id, Patient patient, Doctor doctor, LocalDateTime visitDateTime, String diagnosis, String treatment) {
        if (id == null || id.trim().isEmpty()) {
            throw new IllegalArgumentException(s:"Visit ID cannot be null or empty");
        }
        if (patient == null) {
            throw new IllegalArgumentException(s:"Patient cannot be null");
        }
        if (doctor == null) {
            throw new IllegalArgumentException(s:"Doctor cannot be null");
        }
        if (visitDateTime == null) {
            throw new IllegalArgumentException(s:"Visit date/time cannot be null");
        }

        this.id = id;
        this.patient = patient;
        this.doctor = doctor;
        this.visitDateTime = visitDateTime;
        this.diagnosis = diagnosis;
        this.treatment = treatment;

        logger.info("VisitRecord created: " + id);
    }
}
```

LogTestUtil.JAVA

```
import org.apache.logging.log4j.core.Appender;
import org.apache.logging.log4j.core.LogEvent;
import org.apache.logging.log4j.core.Logger;
import org.apache.logging.log4j.core.appender.AbstractAppender;
import org.apache.logging.log4j.core.layout.PatternLayout;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.core.LoggerContext;
import java.util.ArrayList;
import java.util.List;
```

Юнит тестүүдэд
Log4j лог
мессежүүдийг барих,
шалгах
зориулалттай туслах
класс.

```
public class LogTestUtil {

    private static final List<String> logMessages = new ArrayList<>();
    private static Appender testAppender;

    public static void setupLogCapturing(String loggerName) {
        logMessages.clear();

        LoggerContext context = (LoggerContext) LogManager.getContext(currentContext:false);
        Logger logger = context.getLogger(loggerName);

        testAppender = new AbstractAppender("TestAppender", null,
            PatternLayout.newBuilder().withPattern(pattern:"%m").build(), false) {
            @Override
            public void append(LogEvent event) {
                logMessages.add(event.getMessage().getFormattedMessage());
            }
        };
        testAppender.start();
        logger.addAppender(testAppender);
    }

    public static void tearDownLogCapturing(String loggerName) {
        LoggerContext context = (LoggerContext) LogManager.getContext(currentContext:false);
        Logger logger = context.getLogger(loggerName);

        if (testAppender != null) {
            logger.removeAppender(testAppender);
            testAppender.stop();
        }
    }

    public static List<String> getLogMessages() {
        return logMessages;
    }

    public static boolean containsLog(String expectedMessage) {
        return logMessages.stream().anyMatch(msg -> msg.contains(expectedMessage));
    }
}
```

Бүх классын функцуудад тест хийхэдгээр класс байгаа.

✓ test

✓ java \ hospital

✓ AppointmentTest.java

✓ DoctorTest.java

✓ PatientTest.java

✓ VisitRecordTest.java

✓ 8/8

6.0s

✓ medical-system 61ms

✓ { } hospital 61ms

> ✓ AppointmentTest 40ms

✓ DoctorTest 6.0ms

✓ testSetSpecializationValid() 2.0ms

✓ testSetSpecializationInvalid() 4.0ms

✓ PatientTest 6.0ms

✓ testSetAgeValid() 3.0ms

✓ testSetAgeInvalid() 3.0ms

✓ VisitRecordTest 9.0ms

✓ testVisitRecordCreation() 5.0ms

✓ testInvalidVisitRecordId() 4.0ms

ГАРАЛТ

```
[INFO] Running hospital.VisitRecordTest
[22:24:23] [main] INFO hospital.User - User created: id=P003, name=Dorjoo
[22:24:23] [main] INFO hospital.Patient - Patient age updated: 40
[22:24:23] [main] INFO hospital.Patient - Patient created: id=P003, name=Dorjoo
[22:24:23] [main] INFO hospital.User - User created: id=D003, name=Suvdaa
[22:24:23] [main] INFO hospital.Doctor - Doctor specialization updated: Diagnostician
[22:24:23] [main] INFO hospital.Doctor - Doctor created: id=D003, name=Suvdaa, specialization=Diagnostician
[22:24:23] [main] INFO hospital.VisitRecord - VisitRecord created: V001
[22:24:23] [main] INFO hospital.VisitRecordTest - VisitRecord test setup completed.
[22:24:23] [main] INFO hospital.VisitRecordTest - VisitRecord creation test passed.
[22:24:23] [main] INFO hospital.User - User created: id=P003, name=Dorjoo
[22:24:23] [main] INFO hospital.Patient - Patient age updated: 40
[22:24:23] [main] INFO hospital.Patient - Patient created: id=P003, name=Dorjoo
[22:24:23] [main] INFO hospital.User - User created: id=D003, name=Suvdaa
[22:24:23] [main] INFO hospital.Doctor - Doctor specialization updated: Diagnostician
[22:24:23] [main] INFO hospital.Doctor - Doctor created: id=D003, name=Suvdaa, specialization=Diagnostician
[22:24:23] [main] INFO hospital.VisitRecord - VisitRecord created: V001
[22:24:23] [main] INFO hospital.VisitRecordTest - VisitRecord test setup completed.
[22:24:23] [main] ERROR hospital.VisitRecordTest - Exception caught: Visit ID cannot be null or empty
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.013 s -- in hospital.VisitRecordTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.588 s
```

```
Test Runner for Java
✓ testAppointmentCreation() $(symbol-class) AppointmentTest < $(symbol-name)
✓ testInvalidAppointmentId() $(symbol-class) AppointmentTest < $(symbol-name)
✓ testInvalidVisitRecordId() $(symbol-class) VisitRecordTest < $(symbol-name)
✓ testVisitRecordCreation() $(symbol-class) VisitRecordTest < $(symbol-name)
✓ testSetAgeInvalid() $(symbol-class) PatientTest < $(symbol-name)
✓ testSetAgeValid() $(symbol-class) PatientTest < $(symbol-name)
✓ testSetSpecializationInvalid() $(symbol-class) DoctorTest < $(symbol-name)
✓ testSetSpecializationValid() $(symbol-class) DoctorTest < $(symbol-name)
```

Тест алдаагүй ажиллаж байгаа гаралт

mvn clean test команд өгсөн гаралт

Лог файл

```
[2025-06-11 18:23:33] INFO User:20 - User created: id=D001, name=Boldoo
[2025-06-11 18:23:33] INFO Doctor:27 - Doctor specialization updated: Cardiologist
[2025-06-11 18:23:33] INFO Doctor:13 - Doctor created: id=D001, name=Boldoo, specialization=Cardiologist
[2025-06-11 18:23:33] INFO DoctorTest:16 - Doctor test setup completed.
[2025-06-11 18:23:33] INFO Doctor:27 - Doctor specialization updated: Pediatrician
[2025-06-11 18:23:33] INFO DoctorTest:23 - Doctor specialization updated successfully.
[2025-06-11 18:23:33] INFO User:20 - User created: id=P003, name=Dorjoo
[2025-06-11 18:23:33] INFO Patient:27 - Patient age updated: 40
[2025-06-11 18:23:33] INFO Patient:13 - Patient created: id=P003, name=Dorjoo
[2025-06-11 18:23:33] INFO User:20 - User created: id=D003, name=Suvdaa
[2025-06-11 18:23:33] INFO Doctor:27 - Doctor specialization updated: Diagnostician
[2025-06-11 18:23:33] INFO Doctor:13 - Doctor created: id=D003, name=Suvdaa, specialization=Diagnostician
[2025-06-11 18:23:33] INFO VisitRecord:42 - VisitRecord created: V001
[2025-06-11 18:23:33] INFO VisitRecordTest:21 - VisitRecord test setup completed.
[2025-06-11 18:23:33] INFO VisitRecordTest:31 - VisitRecord creation test passed.
[2025-06-11 18:23:33] INFO User:20 - User created: id=P003, name=Dorjoo
[2025-06-11 18:23:33] INFO Patient:27 - Patient age updated: 40
[2025-06-11 18:23:33] INFO Patient:13 - Patient created: id=P003, name=Dorjoo
[2025-06-11 18:23:33] INFO User:20 - User created: id=D003, name=Suvdaa
[2025-06-11 18:23:33] INFO Doctor:27 - Doctor specialization updated: Diagnostician
[2025-06-11 18:23:33] INFO Doctor:13 - Doctor created: id=D003, name=Suvdaa, specialization=Diagnostician
[2025-06-11 18:23:33] INFO VisitRecord:42 - VisitRecord created: V001
[2025-06-11 18:23:33] INFO VisitRecordTest:21 - VisitRecord test setup completed.
```

Сурч мэдсэн зүйлс

JAVA хэлний мэдлэгээ өргөжүүлж
мини төсөл хийсэн.

TTD буюу Test-Driven Development
аргачлалыг судлан ашиглаж анхан
шатны мэдлэгтэй болсон.

Цахим орчинд өөрийн хийсэн
зүйлсийг GITHUB ашиглан
хадгалж, хуваалцаж сурсан.

JAVA хэл дээр лог бичих зорилгоор
Log4j2 ашигласан энэ нь JAVA дээрх
хүчирхэг, өндөр гүйцэтгэлтэй
фрэймворк юм.



Ашигласан материал:

- [Odoo.com](https://www.odoo.com) - Төсөлийн санаа архитектур
- [JAVA version17](https://www.oracle.com/uk/technologies/javase-downloads/) - Програмчлалын хэл
- [MAVEN](https://maven.apache.org/) - Автоматжуулалтын хэрэгсэл
- [Log4j2](https://log4j.apache.org/) - Лог бичих систем
- [JUnit 5](https://junit.org/junit5/) - Тестийн фрэймворк
- [VS code](https://code.visualstudio.com/) - IDE
- [Github](https://github.com/) - Код хадгалах, хуваалцах
- [Teams](https://teams.microsoft.com/) - Даалгавар авах, явуулах



**Анхаарал тавьсанд
баярлалаа!!!**