1. Singletone

В класі DBMS:

```
private static DBMS instance;

// For Singleton pattern

public static DBMS getInstance() {

    if (instance == null) {

        instance = new DBMS();

    }

    return instance;

}
```

Цей метод використовується в класі UiController

```
public void initialize() {

    this.dbService = new ClinicDbService(DBMS.getInstance());

   .........

}
```

2. Builder

```
TitleBuilder via Builder pattern

public class TitleBuilder {


    private final Admission admission;


    private Patient patient = null;


    public TitleBuilder(Admission admission) {

        this.admission = admission;

    }
    public TitleBuilder setPatient(Patient patient) {
        this.patient = patient;
        return this;
    }

    public String buildTitle() {
        return build();
    }

    private String build() {
        return (patient != null)
                ? patient.getName() + ", tel: " +
patient.getPhoneNumber() + "; " + admission.getDescription()
                : "Reserved time";
    }
}
```

Приклад використання в класі UiController, title для admission створюється так:

```
String title = new
TitleBuilder(admission).setPatient(currentPatient).buildTitle
();
```

3. Decorator

Був реалізований інтерфейс для IndexService, IndexSeviceWithLogs

```
package com.somihmih.er.indexservice;

public interface Indexes {
    void saveToFile();

    void loadIndexes();

    void recreateIndexFile(Index[] indices);

    void addIndex(Index index);

    int getNewId();

    Index getNewIndex();

    int getPosition(int id);

    Index getIndexFor(int id);
}

public class IndexService implements Indexes {
………}
IndexServiceWithLogs via Decorator pattern
public class IndexServiceWithLogs implements Indexes {
………}
```

4. Template Method

```
abstract class PrintableEntity {

    abstract String getEntityType();

    abstract String[] getValuesToPrint();

    abstract boolean isDeleted();

    Template Method
    @Override
    public String toString() {
        String values = "";
        for (String value : getValuesToPrint()) {
            values += value + ", ";
        }
        values = values.substring(0, values.length() - 2);

        String isDeleted = (isDeleted()) ? ", DELETED" : "";

        return getEntityType() + ": (" + values + isDeleted +
")";
    }
}
```

Це шаблон виведення полів різних entity(admission or patient). Тобто тут цей метод відрізнятиметься для patient or admission лише однією частиною

```
}
```

Приклад використання для patient

```
public class Patient extends PrintableEntity implements
Entity {
@Override
String getEntityType() {
    return "Patient";
}

@Override
String[] getValuesToPrint() {
    return new String[] {
            "id:" + id,
            "name=" + name,
            "phoneNumber=" + phoneNumber,
            "addmissionId=" + admissionId
    };
}
```

Приклад використання для admission

```
public class Admission extends PrintableEntity implements
Entity{

@Override
String getEntityType() {
    return "Admission";
}

@Override
```

```
String[] getValuesToPrint() {
    return new String[] {
            "id:" + id,
            "date:'" + date,
            "price:" + patientId,
            "nextAdId:" + nextAdId
    };
}
```

5. Prototype

```
Example of pattern "Prototype"
@Override
public Admission getClone() {
    Admission admission = new Admission(id, date, patientId,
nextAdId, deleted);
    admission.setDescription(description);
```

6. Simple Factory pattern

```
public class DBMS {
    public static final String PATIENTS_INDEX_SERVICE =
"PATIENTS";
    public static final String ADMISSIONS_INDEX_SERVICE =
"ADMISSION";

    private static DBMS instance;
```

```
// For Singleton pattern
    public static DBMS getInstance() {
        if (instance == null) {
            instance = new DBMS();
        }
        return instance;
    }

    public static final String ADMISSIONS = "./dbfiles/
addmissions";
    public static final String PATIENTS = "./dbfiles/
patients";
    public static final String ADMISSION_INDEXES = "./
dbfiles/addmissionIndexes";
    public static final String PATIENTS_INDEXES = "./dbfiles/
patientsIndexes";



public static Indexes createIndexService(String serviceName)
{
    if (serviceName.equals("PATIENTS")) {
        return new IndexServiceWithLogs(new
IndexService(PATIENTS_INDEXES), "Patients");
    } else if (serviceName.equals("ADMISSION")) {
        return new IndexService(ADMISSION_INDEXES);
    }
    // Other
    return null; }
```

7. Iterator

Клас IteratorForFilteredPatients для пошуку, проходження по переліку всіх пацієнтів, що шукаються за певним параметром.

```
public class IteratorForFilteredPatients implements
Iterator<Patient> {

    private final String maskName;
    private final String maskPhone;
    private final Patient[] patients;
    private int currentIndex = 0;

    public IteratorForFilteredPatients(String maskName,
String maskPhone, Patient[] patients) {
        this.maskName = maskName.toLowerCase();
        this.maskPhone = maskPhone.toLowerCase();
        this.patients = patients;
        moveToNextValid();
    }

    @Override
    public boolean hasNext() {
        return currentIndex < patients.length;
    }

    @Override
    public Patient next() {
        if (!hasNext()) {
            throw new NoSuchElementException();
        }
        Patient patient = patients[currentIndex];
        currentIndex++;
        moveToNextValid();
        return patient;
    }

    private void moveToNextValid() {
        while (currentIndex < patients.length) {
            Patient patient = patients[currentIndex];
            String lowerCaseName =
patient.getName().toLowerCase();
            String lowerCasePhone =
patient.getPhoneNumber().toLowerCase();

            boolean isValidByName = maskName.isEmpty() ||
lowerCaseName.contains(maskName);
            boolean isValidByPhone = maskPhone.isEmpty() ||
lowerCasePhone.contains(maskPhone);

            if (isValidByName && isValidByPhone) {
                break;
            }
            currentIndex++;
        }
    }
}
```

Використовується в класі UiController

```
private void updatePatientsList() {
    patientList.clear();
    String maskName = byName.getText();
    String maskPhone = byPhone.getText();

    // 7. Iterator pattern
    IteratorForFilteredPatients iterator = new
IteratorForFilteredPatients(maskName, maskPhone, patients);

    while (iterator.hasNext()) {
        Patient patient = iterator.next();
        patientList.add(patient);
        System.out.println("fillPatientsList: " + patient);
    }
}
```