

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**З В І Т**

**Лабораторна робота №4  
з дисципліни  
«Комп'ютерні системи  
штучного інтелекту»**

Виконавець:

студент групи КІ-22м

Косей М.П.

Керівник:

викладач

Саяпін В.Г.

2023

## Лабораторна робота №3

**Тема:** Структура експертної системи

**Мета:** Одержані уміння та закріпити навички роботи з певним інтегрованим середовищем для логічного програмування мовою PROLOG або іншими мовами або засобами

**Завдання:** Відповідно до узагальненої структури експертних систем визначити основні змістовні блоки системи та розробити власну базу знань

### ХІД РОБОТИ

1) Ознайомитись з теоретичними відомостями до лабораторної роботи.

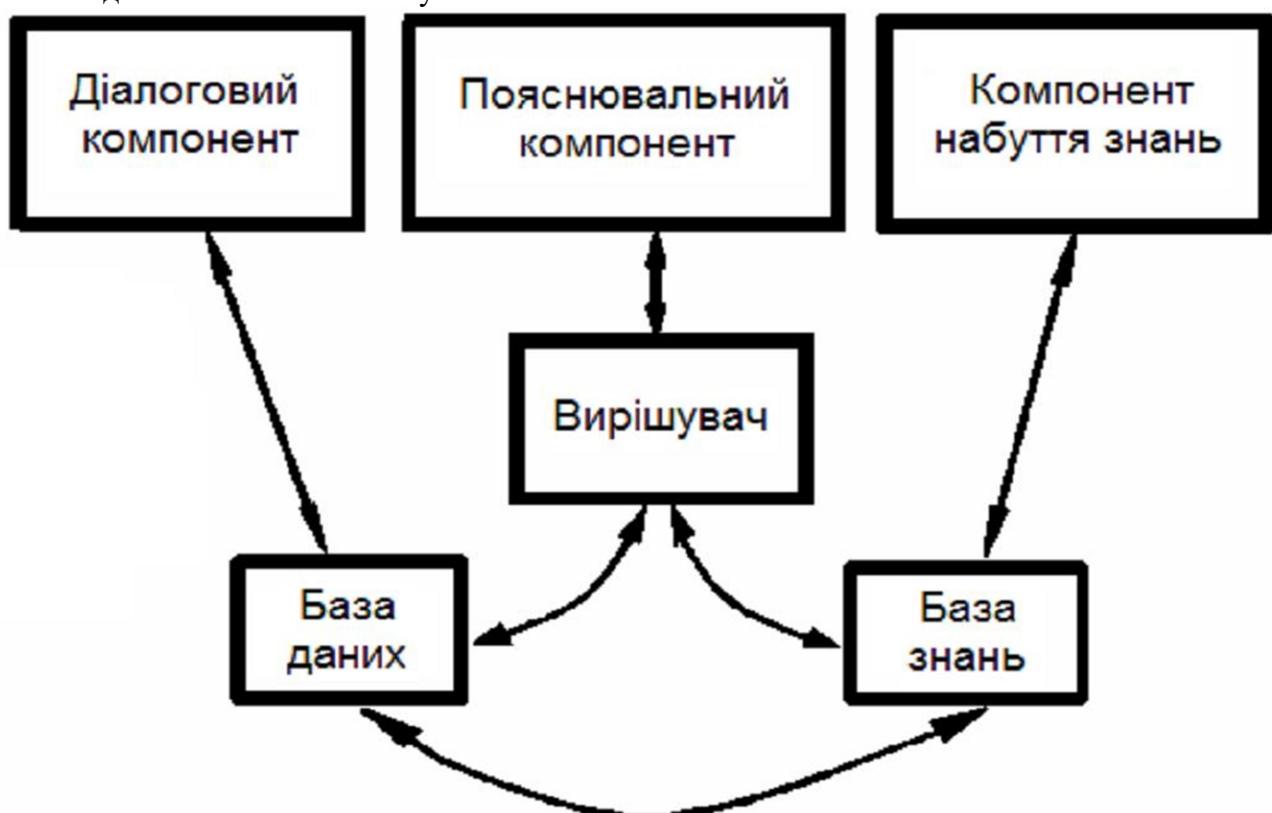
Експертні системи - це програмне забезпечення (ПЗ), що використовує штучний інтелект для вирішення проблем, що вимагають експертного знання в певній області.

Вони зазвичай використовують базу знань, яка містить інформацію про проблему та її рішення, а також правила, за якими програма визначає, яке рішення найкраще підходить для конкретної ситуації.

Експертні системи можуть бути використані в різних галузях, наприклад, в медицині для діагностики захворювань, в фінансах для рішення інвестиційних питань, в інженерії для проектування нових систем та багатьох інших.

Типова статична експертна система (ЕС) складається з наступних основних компонентів (рисунок 4.1.):

- вирішувача (інтерпретатора);
- робочої пам'яті (РП), яка ще називається базою даних (БД);
- бази знань (БЗ);
- компонентів набуття знань;
- пояснювального компоненту;
- діалогового компоненту.



**База даних** призначена для зберігання вихідних та проміжних даних задачі, що вирішується в даний момент. Цей термін схожий за назвою, але не за значенням з терміном, що використовується в інформаційно-пошукових системах (ІПС) і системах управління базами даних (СУБД) для позначення всіх даних (в першу чергу довгострокових), що

зберігаються в системі.

**База знань (БЗ)** в ЕС призначена для зберігання довгострокових даних, що описують дану область (а не поточних даних), і правил, що описують доцільні перетворення даних цієї області.

**Вирішувач**, використовуючи початкові дані з робочої пам'яті і знання з БЗ, формує послідовність правил, які, при застосуванні до початкових даних, приводять до вирішення задачі.

**Компонент набуття знань** автоматизує процес наповнення ЕС знаннями, що здійснюється користувачем-експертом.

**Пояснювальний компонент** пояснює, як система отримала вирішення задачі (або чому вона не отримала розв'язку) і які знання вона при цьому використовувала, що полегшує експерту тестування системи і підвищує довіру користувача до одержаного результату.

Діалоговий компонент орієнтований на організацію дружнього спілкування з користувачем як в ході вирішення задач, так і в процесі придбання знань і пояснення результатів роботи.

У розробці ЕС беруть участь представники наступних спеціальностей:

- експерт з проблемної області, задачі якої вирішує ЕС
- інженер зі знань — фахівець з розробки ЕС (технологію (методи), що ним використовуються, називають технологією (методами) інженерії знань);
- програміст з розробки інструментальних засобів (ІЗ), що призначені для прискорення розробки ЕС.

Експертні системи можуть бути використані в різних галузях, наприклад, в медицині для діагностики захворювань, в фінансах для рішення інвестиційних питань, в інженерії для проектування нових систем та багатьох інших.

Однією з переваг експертних систем є те, що вони можуть бути використані для автоматизації складних процесів та забезпечення консистентності прийнятих рішень. Крім того, вони можуть бути корисні для збільшення продуктивності та зниження витрат, оскільки можуть замінити людський фаховий досвід у багатьох сферах.

Однак, експертні системи мають свої обмеження. Вони можуть бути неефективні при роботі зі складними проблемами, які вимагають значної кількості знань та досвіду. Крім того, їх ефективність залежить від якості бази знань та правил, що визначають, як програма діє у різних ситуаціях.

У майбутньому, з розвитком штучного інтелекту та машинного навчання, можуть з'явитися нові підходи до розробки експертних систем, що дозволять їм бути більш ефективними та точними у вирішенні проблем.

Існує багато різних експертних систем, які використовуються в різних сферах діяльності. Ось декілька прикладів:

- **MYCIN**: експертна система, яка використовується в медицині для діагностики та лікування інфекцій крові. Вона була розроблена у 1970-х роках та досі залишається однією з найбільш відомих експертних систем;
- **DENDRAL**: експертна система, яка використовується в хімії для ідентифікації складних органічних молекул. Вона була розроблена у 1960-х роках та вважається однією з перших експертних систем;
- **XCON**: експертна система, яка використовується виробничому процесі для автоматизації виробничих процесів. Вона була розроблена у 1980-х роках та досі залишається однією з найбільш відомих експертних систем у цій сфері;
- **Prolog Expert System Shell (PESS)**: експертна система, яка використовується для розв'язання різних завдань, включаючи логічне програмування та штучний інтелект. Вона була розроблена у 1990-х роках та досі використовується в різних сферах діяльності.

Існує багато фреймворків для створення експертних систем.

Ось декілька з них:

- **Drools**: відкритий фреймворк на базі Java, який надає правила для виконання бізнес-логіки;

- **CLIPS**: система з розробки правил для Windows та UNIX, яка надає інтеграцію з C та C++;
- **Jess**: фреймворк на базі Java, який надає правила та інструменти для створення експертних систем;
- **Pyke**: фреймворк з відкритим кодом, який надає правила для Python та використовує логічне програмування.

## 2) Розробка експертної системи.

Розробка експертної системи виконується згідно варіанту 8 – «Визначення типу автомашини».

Для розробки використовується мова програмування **JAVA** та фреймворк **Drools**

**Drools** є відкритою системою з активною підтримкою та великою спільнотою розробників (<https://www.drools.org/>) остання версія від 23.03.2023р.).

The screenshot shows the official Drools website homepage. At the top, there's a banner announcing the release of Drools 8.36.0.Final on March 23, 2023. Below this, a brief description of what Drools is (a Business Rules Management System) is provided. To the right, there's a green box titled "Try Drools" with a "Download 8.36.0.Final" button. Below it, instructions for running examples are given, along with a note that Java™ is required. Further down, there's a "Get started" section with a "User guide 8.36.0.Final" link. On the left side of the main content area, there's a screenshot of the Drools Workbench interface, showing a "Patient.java - Data Objects" model with fields like Identifier, Label, and Type, and a "general properties" panel for 'InsuranceCompany'.

### Основні концепції Drools:

- **Факти (Facts)** - представляють дані, які служать вхідними для правил.
- **Робоча пам'ять (Working Memory)** - сховище з фактами, де вони використовуються для відповідності шаблону та можуть бути змінені, вставлені та видалені.
- **Правило (Rule)** - представляє одне правило, яке асоціює факти з відповідними діями. Його можна написати мовою правил Drools у файлах .drl або як таблицю рішень в електронній таблиці Excel.
- **Сесія знань (Knowledge Session)** - вона містить всі ресурси, необхідні для виконання правил; всі факти вставляються в сесію, а потім запускаються відповідні правила.
- **База знань (Knowledge Base)** - представляє знання в екосистемі Drools, вона містить інформацію про ресурси, де знаходяться правила, а також створює сесію знань.
- **Модуль (Module)** - модуль містить кілька баз знань, які можуть містити різні сесії.

Ця експертна система призначена для визначення типу автомашини.

Система буде задавати питання щодо характеристик автомашини та відповідно до відповідей визначати тип автомашини.

### Характеристики автомашини

Серед характеристик автомашини, які впливають на визначення типу автомашини є:

- Розмір автомобіля
- Вартість автомобіля
- Кількість дверей
- Кількість місць

- Кількість колес
- Тип двигуна

Характеристики реалізовується за допомогою коду

```
public class Car { no usages new*
    private String size; 2 usages
    private double cost; 2 usages
    private int doors; 2 usages
    private int seats; 2 usages
    private int wheels; 2 usages
    private String engineType; 2 usages
    private String type; 2 usages

    public String getSize() { return size; }

    public void setSize(String size) { this.size = size; }

    public double getCost() { return cost; }

    public void setCost(double cost) { this.cost = cost; }

    public int getDoors() { return doors; }

    public void setDoors(int doors) { this.doors = doors; }

    public int getSeats() { return seats; }

    public void setSeats(int seats) { this.seats = seats; }

    public int getWheels() { return wheels; }

    public void setWheels(int wheels) { this.wheels = wheels; }

    public String getEngineType() { return engineType; }

    public void setEngineType(String engineType) { this.engineType = engineType; }

    public String getType() { return type; }

    public void setType(String type) { this.type = type; }
}
```

### **Визначення типу автомобінни**

Система визначення типу автомобінни розбита на кілька категорій:

- Легкові автомобілі
- Вантажні автомобілі
- Автобуси
- Мотоцикли

За відповідність характеристик автомашини до кожної з цих категорій, система буде визначати тип автомашини згідно з правилами.

Правило включає конструкцію **When-Then**, де розділ **When** перелічує умову, яку необхідно перевірити, а розділ **Then** перелічує дію, яку необхідно виконати, якщо умова виконується.

```
//Cars
rule "Легкові автомобілі"
when
    $size : Car(size == "small" || size == "medium") &&
    $cost : Car(cost > 5000 && cost < 20000) &&
    $doors : Car(doors == 2 || doors == 4) &&
    $seats : Car(seats == 2 || seats == 4) &&
    $wheels : Car(wheels == 4) &&
    $engineType : Car(engineType == "gasoline" || engineType == "diesel")
then
    System.out.println("Це легковий автомобіль");
end
//Trucks
rule "Вантажні автомобілі"
when
    $size : Car(size == "large") &&
    $cost : Car(cost > 15000 && cost < 50000) &&
    $doors : Car(doors == 2) &&
    $seats : Car(seats == 2) &&
    $wheels : Car(wheels == 6) &&
    $engineType : Car(engineType == "diesel")
then
    System.out.println("Це вантажний автомобіль");
end

//Buses
rule "Автобуси"
when
    $size : Car(size == "large") &&
    $cost : Car(cost > 25000 && cost < 100000) &&
    $doors : Car(doors == 2) &&
    $seats : Car(seats > 10 && seats < 50) &&
    $wheels : Car(wheels == 6) &&
    $engineType : Car(engineType == "diesel")
then
    System.out.println("Це автобус");
```

```

//Motorcycles
rule "Мотоцикли"
when
    $size : Car(size == "small") &&
    $cost : Car(cost > 1000 && cost < 20000) &&
    $seats : Car(seats == 1 || seats == 2) &&
    $wheels : Car(wheels == 2) &&
    $engineType : Car(engineType == "gasoline")
then
    System.out.println("Це мотоцикл");
end

```

Код для завантаження правил створення сесії Drools.

```

public class DroolsBeanFactory { 2 usages new*
    private KieServices kieServices = KieServices.Factory.get(); 4 usages

    private void getKieRepository() { 1 usage new*
        final KieRepository kieRepository = kieServices.getRepository();
        kieRepository.addKieModule(kieRepository::getDefaultReleaseId);
    }

    public KieSession getKieSession() { 1 usage new*
        getKieRepository();
        KieFileSystem kieFileSystem = kieServices.newKieFileSystem();

        kieFileSystem.write(ResourceFactory.newClassPathResource("ua/edu/knu/drools/rules/rules-car-expert-system.drl"));

        KieBuilder kb = kieServices.newKieBuilder(kieFileSystem);
        kb.buildAll();
        KieModule kieModule = kb.getKieModule();

        KieContainer kContainer = kieServices.newKieContainer(kieModule.getReleaseId());

        return kContainer.newKieSession();
    }
}

```

Код для визначення типу автомашини.

```

public class App { no usages new*
    public static void main(String[] args) throws IOException { no usages new*
        // Створення нової сесії Drools
        KieSession ksession = new DroolsBeanFactory().getKieSession();

        // Створення нового об'єкту Car для запитань та відповідей
        Car car = new Car();

        // Задання питань про характеристики автомобіля та запис в об'єкт Car
        System.out.println("Який розмір вашого автомобіля? (small, medium, large)");
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        String size = reader.readLine();
        car.setSize(size);
    }
}

```

```

System.out.println("Яка вартість вашого автомобіля? (введіть число)");
double cost = Double.parseDouble(reader.readLine());
car.setCost(cost);

System.out.println("Скільки дверей у вашому автомобілі?");
int doors = Integer.parseInt(reader.readLine());
car.setDoors(doors);

System.out.println("Скільки місць у вашому автомобілі?");
int seats = Integer.parseInt(reader.readLine());
car.setSeats(seats);

System.out.println("Скільки коліс у вашому автомобілі?");
int wheels = Integer.parseInt(reader.readLine());
car.setWheels(wheels);

System.out.println("Який тип двигуна у вашому автомобілі? (gasoline, diesel)");
String engineType = reader.readLine();
car.setEngineType(engineType);

// Виконання правил в режимі експерта та визначення типу автомобіля
ksession.insert(car);
ksession.fireAllRules();

// Закриття сесії Drools
ksession.dispose();
}

```

Приклади виконання:

```

C:\Users\Max11mus\.jdks\azul-11.0.12\bin\java.exe ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Який розмір вашого автомобіля? (small, medium, large)
small
Яка вартість вашого автомобіля? (введіть число)
10000
Скільки дверей у вашому автомобілі?
0
Скільки місць у вашому автомобілі?
2
Скільки коліс у вашому автомобілі?
2
Який тип двигуна у вашому автомобілі? (gasoline, diesel)
gasoline
Це мотоцикл

Process finished with exit code 0

```

```
C:\Users\Max11mus\.jdks\azul-11.0.12\bin\java.exe ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Який розмір вашого автомобіля? (small, medium, large)
large
Яка вартість вашого автомобіля? (введіть число)
30000
Скільки дверей у вашому автомобілі?
2
Скільки місць у вашому автомобілі?
28
Скільки коліс у вашому автомобілі?
6
Який тип двигуна у вашому автомобілі? (gasoline, diesel)
diesel
Це автобус
```

```
Process finished with exit code 0
```

## **ВИСНОВКИ**

**В результаті виконаної лабораторної роботи розроблена проста експертна система для визначення типу автомашини.**

**Усі матеріали викладені у репозіторії GitHub, за посиланням <https://github.com/Max11mus/Artifition-Intelect-Lab4.git>.**