




Lesson 25

27.07.2023







```
public class Ex1 extends Test {
    public void display() throws IOException {
        System.out.println("Derived");
    }

    public static void main(String[] args) throws IOException {
        Ex1 object = new Ex1();
        object.display();
    }
}

class Test {

    public void display() throws IOException {
        System.out.println("Test");
    }
}
```





```
class Helper {
    private int data;

    private Helper() {
        data = 5;
    }
}

public class Ex2 {
    public static void main(String[] args) {
        Helper help = new Helper();
        System.out.println(help.data);
    }
}
```

```
class Base {
    private int data;

    public Base() {
        data = 5;
    }

    public int getData() {
        return this.data;
    }
}


class Ex3 extends Base {
    private int data;

    public Ex3() {
        data = 6;
    }

    public int getData() {
        return data;
    }


    public static void main(String[] args) {
        Ex3 myData = new Ex3();
        System.out.println(myData.getData());
    }
}
```

```
public static void main(String[] args) {  
    int[] arr = {2, 1, 0};  
    for (int i : arr) {  
        System.out.println(arr[i]);  
    }  
}
```

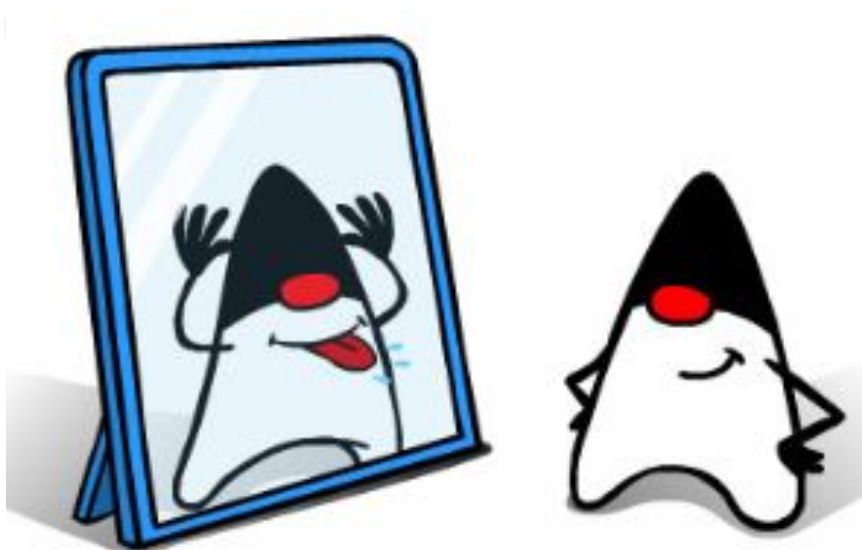


```
public class Ex5 {
    private static void checkData() throws SQLException {
        try {
            throw new SQLException();
        } catch (Exception e) {
            e = null;
            throw e;
        }
    }

    public static void main(String[] args) {
        try {
            checkData(); //Line 17
        } catch (SQLException e) {
            System.out.println("NOT AVAILABLE");
        }
    }
}
```



Рефлексія (від reflexio – звернення назад) – це механізм дослідження даних про програму під час її виконання. Рефлексія Java здійснюється за допомогою Java Reflection API, що складається з класів пакетів `java.lang` і `java.lang.reflect`. В інформатиці рефлексія означає процес, під час якого програма може відстежувати та модифікувати власну структуру та поведінку під час виконання.





Java Reflection API дозволяє отримувати інформацію про конструктори, методи та поля класів і виконувати наступні операції над полями та методами об'єкта/класу :

- визначення класу об'єкта;
- отримання інформації про поля, методи, конструктори та суперкласи;
- отримання інформації про модифікаторів полів та методів;
- створення екземпляра класу, ім'я якого невідоме до моменту виконання програми;
- визначення та зміна значень властивостей об'єкта/класу;
- виклик методів об'єкта/класу.

Визначення властивостей класу

У працюючій програмі для отримання класу необхідно використовувати метод `forName (String className)`.

```
public class ClassDefinition {  
    public static void main(String[] args) throws ClassNotFoundException {  
        Class fooRefl = Class.forName("Lesson24.reflection.Foo");  
        System.out.println(fooRefl.getName());  
    }  
}  
  
class Foo{  
  
    void print(){  
        System.out.println("Class >> Foo.java");  
    }  
}
```



Визначення інтерфейсів класу

Для отримання в режимі run-time списку інтерфейсів, що реалізують класом, необхідно отримати Class і використовувати його метод **getInterfaces()**

```
public class ClassGetInterfaces {  
    public static void main(String[] args) {  
        Class<?>    cls = ArrayList.class;  
        Class<?>[] ifs = cls.getInterfaces();  
  
        System.out.println("List of interfaces\n");  
        for(Class<?> ifc : ifs) {  
            System.out.println (ifc.getName());  
        }  
    }  
}
```

Визначення конструкторів класу

Метод класу **getConstructors()** дозволяє отримати масив відкритих конструкторів типу `java.lang.reflect.Constructor`. Після цього, можна витягувати інформацію про типи параметрів конструктора і винятків, що генеруються :

```
public class ClassGetConstructor {  
    public static void main(String[] args) throws ClassNotFoundException {  
        Class<?> cls = Class.forName("Lesson24.reflection.Baz");  
        Constructor[] constructors = cls.getConstructors();  
        for (Constructor constructor : constructors) {  
            System.out.println(constructor);  
            Class<?>[] params = constructor.getParameterTypes();  
            for (Class<?> param : params) {  
                System.out.println(param.getName());  
            }  
        }  
    }  
}
```

Визначення полів класу

Метод **getFields()** об'єкта Class повертає масив відкритих полів типу `java.lang.reflect.Field`, які можуть бути визначені не тільки в даному класі, але також і в його батьках (суперкласі) або інтерфейсах, реалізованих класом або його батьками. Клас `Field` дозволяє отримати ім'я поля, тип та модифікатори.

```
public class ClassGetFields {  
    public static void main(String[] args) throws ClassNotFoundException {  
        Class<?> cls = Class.forName("Lesson24.reflection.Fee");  
        Field[] fields = cls.getFields();  
        for (Field field : fields) {  
            Class<?> fld = field.getType();  
            System.out.println("Class name : " + field.getName());  
            System.out.println("Class type : " + fld.getName());  
        }  
    }  
}
```



Документування javadoc

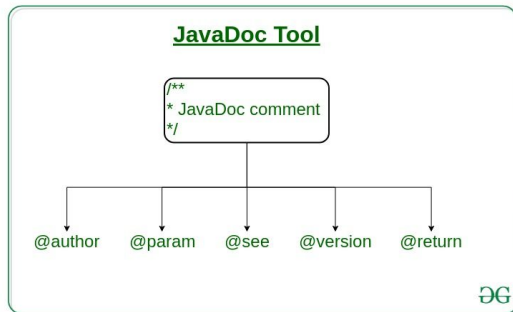
Javadoc - це генератор документації в HTML-форматі з коментарів вихідного коду Java та визначає стандарт для документування класів Java. Для створення доклетів та теглетів, які дозволяють програмісту аналізувати структуру Java-програми, javadoc також надає API. У кожному випадку коментар повинен бути перед документованим елементом.

При написанні коментарів до кодів Java використовують три типи коментарів:

// однорядковий коментар;

/* багаторядковий коментар */

/** Коментування документації */





| Tag | Parameter | Description |
|----------|--------------|--|
| @author | author_name | Describes an author |
| @param | description | provide information about method parameter or the input it takes |
| @see | reference | generate a link to other element of the document |
| @version | version-name | provide version of the class, interface or enum. |
| @return | description | provide the return value |



| Tag | Parameter | Description |
|----------|--------------|--|
| @author | author_name | Describes an author |
| @param | description | provide information about method parameter or the input it takes |
| @see | reference | generate a link to other element of the document |
| @version | version-name | provide version of the class, interface or enum. |
| @return | description | provide the return value |



```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-javadoc-plugin</artifactId>
      <version>3.0.0-M1</version>
      <configuration>
        <stylesheetfile>${basedir}/src/main/javadoc/stylesheet.css</stylesheetfile>
        <show>public</show>
      </configuration>
    </plugin>
  </plugins>
</reporting>
```

```
mvn javadoc:javadoc
```