



# Lesson 11

15.01.2024


```
public class Ex1 {  
    public static void main(String[] args) {  
        Long a = null;  
        long b = 0;  
  
        if (a == b) {  
            System.out.println("==");  
        } else {  
            System.out.println("!=");  
        }  
    }  
}
```

```
public class Ex2 {  
    public static void main(String[] args) {  
        System.out.println(null);  
    }  
}
```




```
public class Ex3 extends Ex3_1 {
    public void print(int d) {
        System.out.println("EX3");
    }

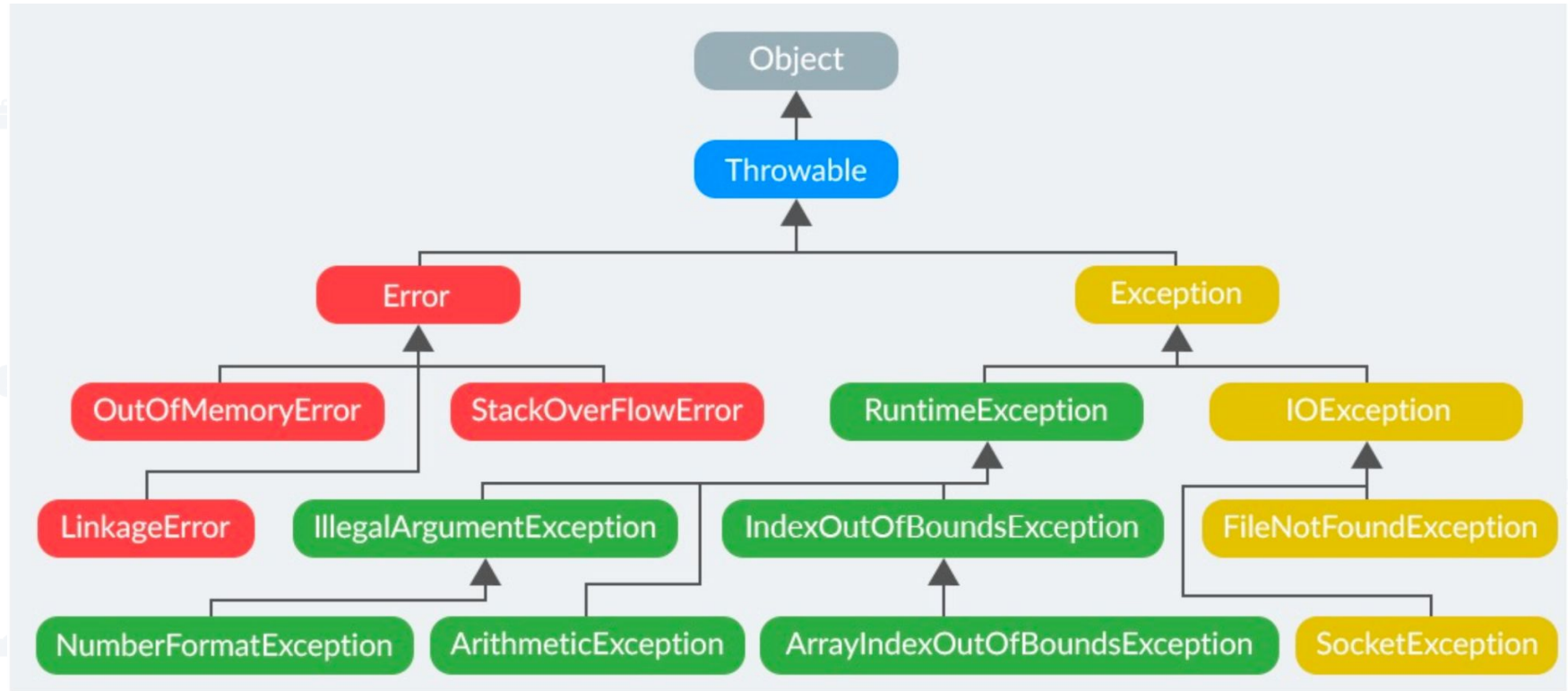
    public static void main(String[] args) {
        Ex3 ex = new Ex3();
        ex.print(1);
        ex.print(2.0);
    }
}

class Ex3_1 {
    public void print(double d) {
        System.out.println("EX3_1");
    }
}
```



```
public class Ex5 {  
    public static void main(String[] args) {  
        TreeSet<String> set = new TreeSet<>();  
        set.add("Java");  
        set.add("The");  
        set.add("Java");  
        set.add("JavaTheBest");  
  
        for (String str : set) {  
            System.out.print(str + " ");  
        }  
        System.out.println("\n");  
    }  
}
```





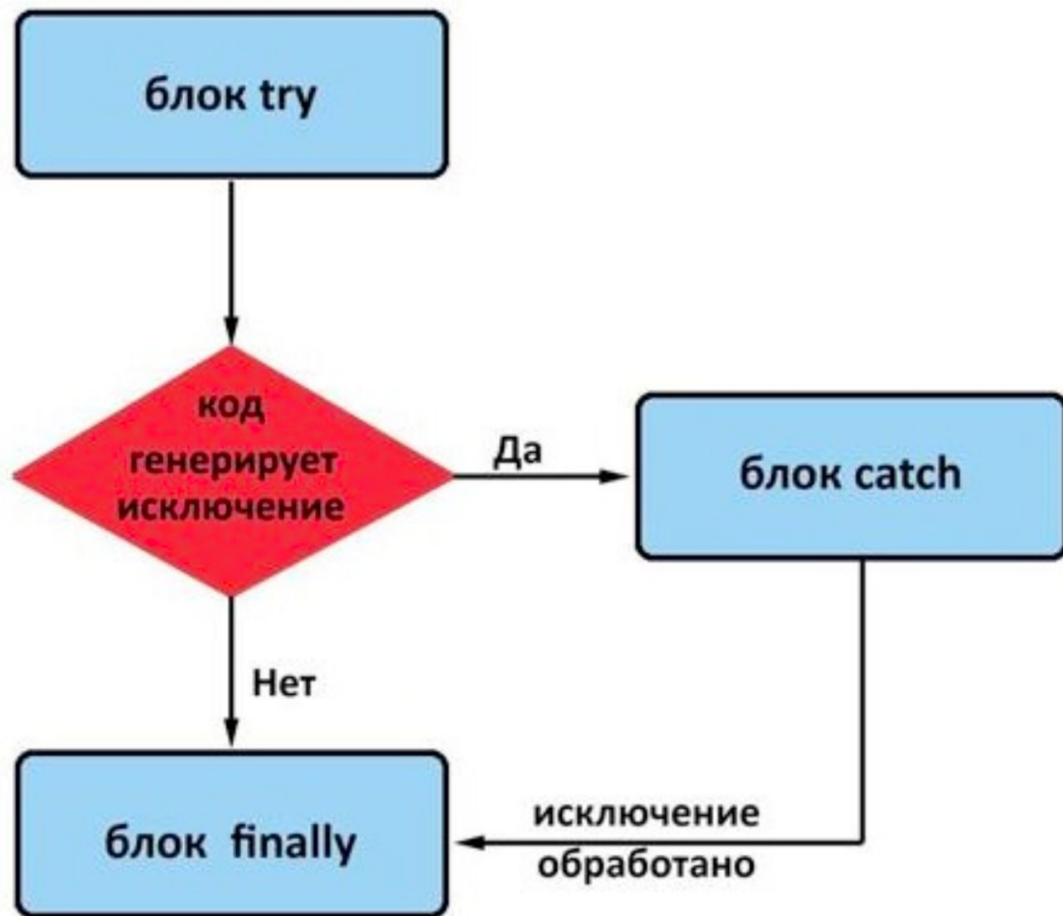
**try** – визначає блок коду, в якому може бути виключено;

**catch** – визначає блок коду, в якому відбувається обробка виключення;

**finally** – визначає код блоку, який не є обов'язковим, але при його наявності виконується в будь-якому випадку незалежно від результатів виконання блоку try.

**throw** – використовується для возбуждення виключення;

**throws** – використовується в сигнатурі методів для попередження, про те, що метод може викинути виключення.





```
public class TryCatch {  
    public static void main(String[] args) {  
        try {  
            System.err.print(" 0");  
            if (true) {  
                throw new RuntimeException();  
            }  
            System.err.print(" 1");  
        } catch (RuntimeException e) {  
            System.err.print(" 2");  
        }  
        System.err.println(" 3");  
    }  
}
```

try + catch + catch + ...

```
public class TryCatchCatch {  
    public static void main(String[] args) {  
        try {  
            throw new Exception();  
        } catch (RuntimeException e) {  
            System.err.println("catch RuntimeException");  
        } catch (Exception e) {  
            System.err.println("catch Exception");  
        } catch (Throwable e) {  
            System.err.println("catch Throwable");  
        }  
        System.err.println("next statement");  
    }  
}
```

```
public class TryFinally {  
    public static void main(String[] args) {  
        try {  
            throw new RuntimeException();  
        } finally {  
            System.err.println("finally");  
        }  
    }  
}
```


try + catch + finally

```
public class TryCatchFinally {  
    public static void main(String[] args) {  
        try {  
            System.err.print(" 0");  
            if (true) {throw new Error();}  
            System.err.print(" 1");  
        } catch(Error e) {  
            System.err.print(" 2");  
        } finally {  
            System.err.print(" 3");  
        }  
        System.err.print(" 4");  
    }  
}
```

## try-with-resources

```
public static void main(String[] args) {
    Scanner scanner = null;
    try {
        scanner = new Scanner(new File( pathname: "test.txt"));
        while (scanner.hasNext()) {
            System.out.println(scanner.nextLine());
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } finally {
        if (scanner != null) {
            scanner.close();
        }
    }
}

public static void main(String[] args) {
    try (Scanner scanner = new Scanner(new File( pathname: "test.txt"))) {
        while (scanner.hasNext()) {
            System.out.println(scanner.nextLine());
        }
    } catch (FileNotFoundException fnfe) {
        fnfe.printStackTrace();
    }
}
```



Необхідно розуміти, що

- перевірка виключення **checked** відбувається в момент компіляції (перевірка під час компіляції)
- перехват виключення (catch) відбувається в момент виконання (перевірка виконання)