




Lesson 11


01.06.2023



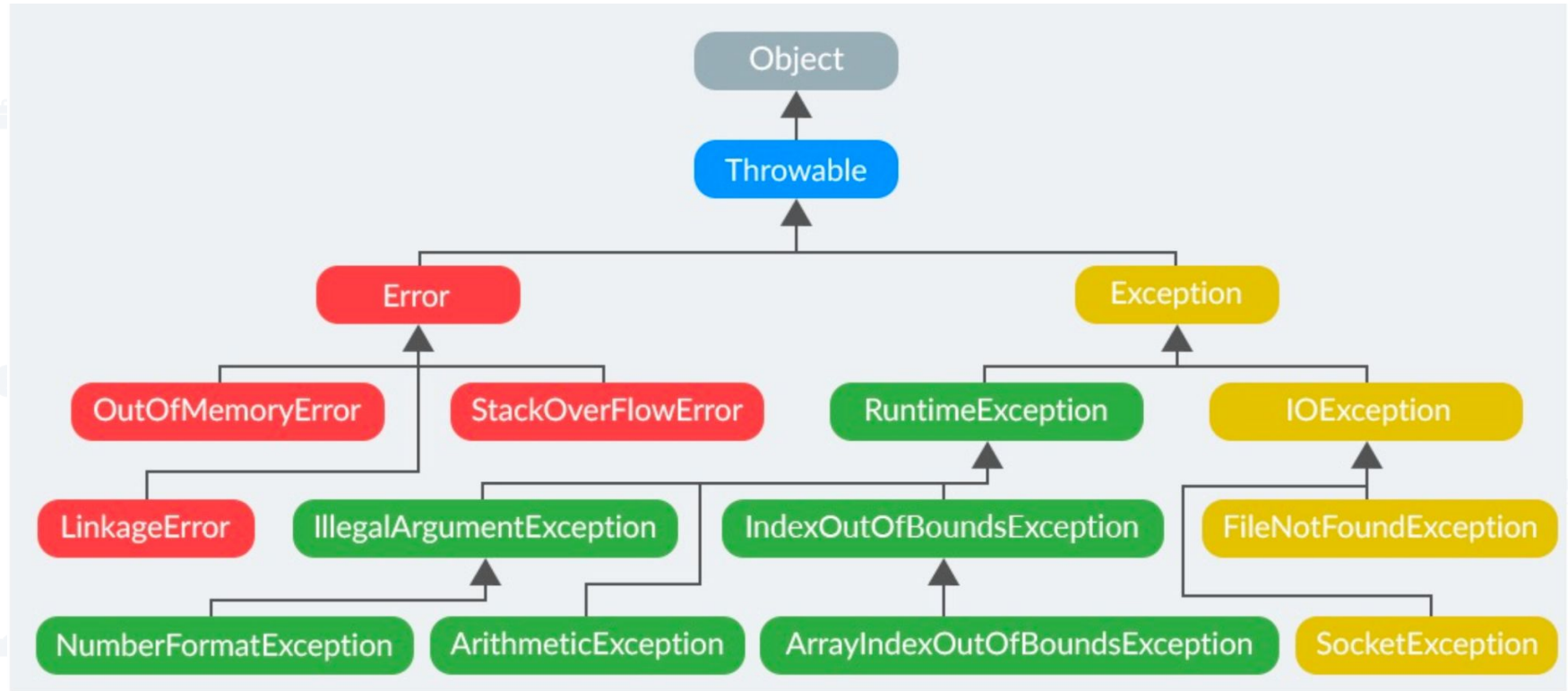
```
public class Ex1 {  
    public static void main(String[] args) {  
        Long a = null;  
        long b = 0;  
  
        if (a == b) {  
            System.out.println("==");  
        } else {  
            System.out.println("!=");  
        }  
    }  
}
```

```
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");  
    }  
}
```



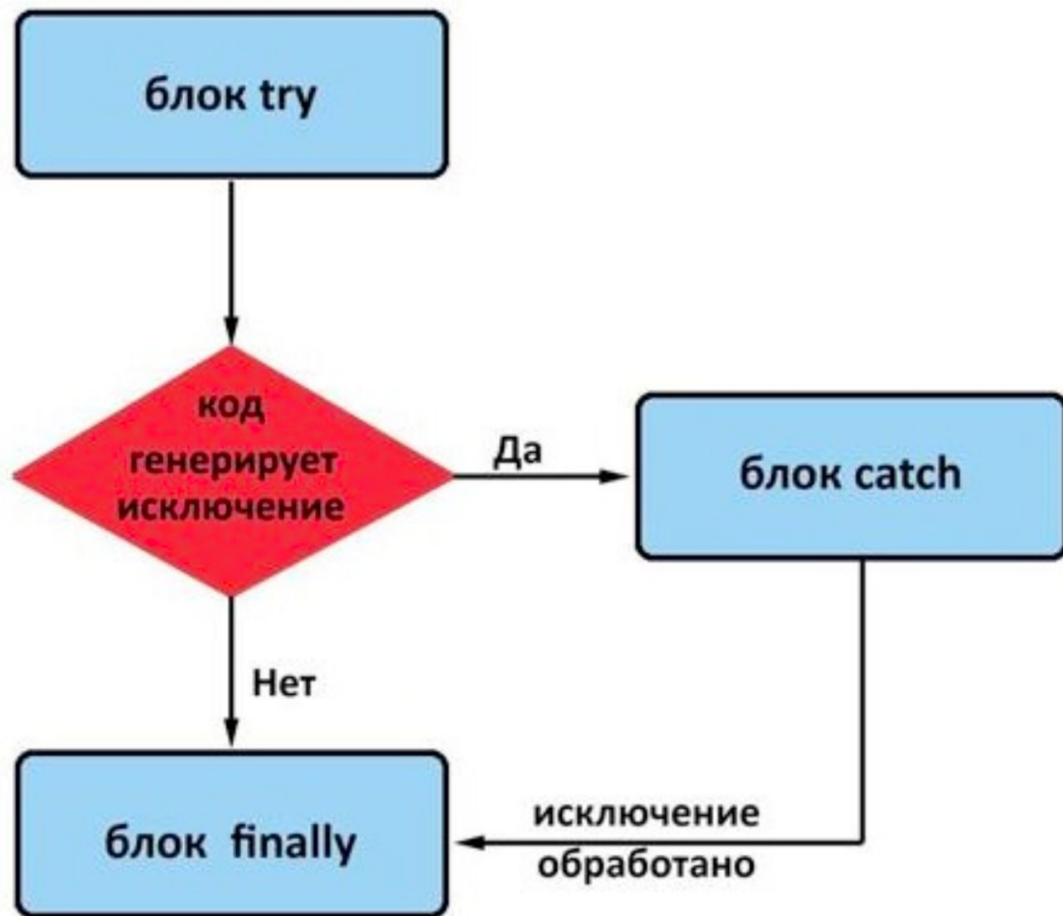
try – визначає блок коду, в якому може бути виключено;

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)



Операції з Set

1. **add()** - додає елемент у Set
2. **remove()** - видаляє елемент із Set
3. **contains()** - визначає, є чи елемент у Set
4. **size()** - повертає розмір Set
5. **clear()** - видаляє всі елементи з колекції
6. **isEmpty()** - повертає true, якщо безліч порожніх, і false, якщо там є хоча 1 елемент



String

У String є дві фундаментальні особливості:


- це незмінний (immutable) клас
- це final клас

Загалом, у класі String не може бути спадкоємців (final), а екземпляри класу не можна змінювати після створення (immutable)

StringBuffer и StringBuilder

Відмінність між String, StringBuilder, StringBuffer:

- Класи StringBuffer і StringBuilder в Java використовуються, коли виникає необхідність внести багато змін у рядок символів.
- На відміну від String, об'єкти типу StringBuffer і StringBuilder можуть бути змінені знову і знову.
- Основна різниця між StringBuffer і StringBuilder в Java полягає в тому, що методи StringBuilder не є безпечними для потоків (несинхронізовані).
- Рекомендується використовувати StringBuilder кожен раз, коли це можливо, що він швидше, ніж StringBuffer в Java.
- Однак, якщо необхідна безпека потоків, кращим варіантом є об'єкти StringBuffer.



Метод **charAt()** повертає символ у вказаній позиції. А **setCharAt()** змінює символ у вказаній позиції

Метод **append()** приєднує підстроку до рядка

Метод **insert()** ставить підстроку у вказану позицію

Метод **reverse()** використовується для інвертування рядків

Метод **delete()** видаляє підстроку, використовуючи вказані позиції

Метод **deleteCharAt()** видаляє символ із зазначеної позиції

Метод **replace()** змінює підстроку у вказаній позиції іншою