



ОНЛАЙН-ОБРАЗОВАНИЕ

32– Handling Exceptions (Часть 2)

Дмитрий Коган



Как меня слышно и видно?



Если нет – напишите, если слышите – смайлик в чат.



Цели :

- Изучим лабиринт **try-catch-finally** до последнего закутка





Начинаем?

Темы экзамена

- ☐ Java Basics
- ☐ Working with Java Data Types
- ☐ Using Operators and Decision Constructs
- ☐ Creating and Using Arrays
- ☐ Using Loop Constructs
- ☐ Working with Methods and Encapsulation
- ☐ Working with Inheritance
- ☐ **Handling Exceptions**
- ☐ Working with Selected classes from the Java API

Подтемы экзамена

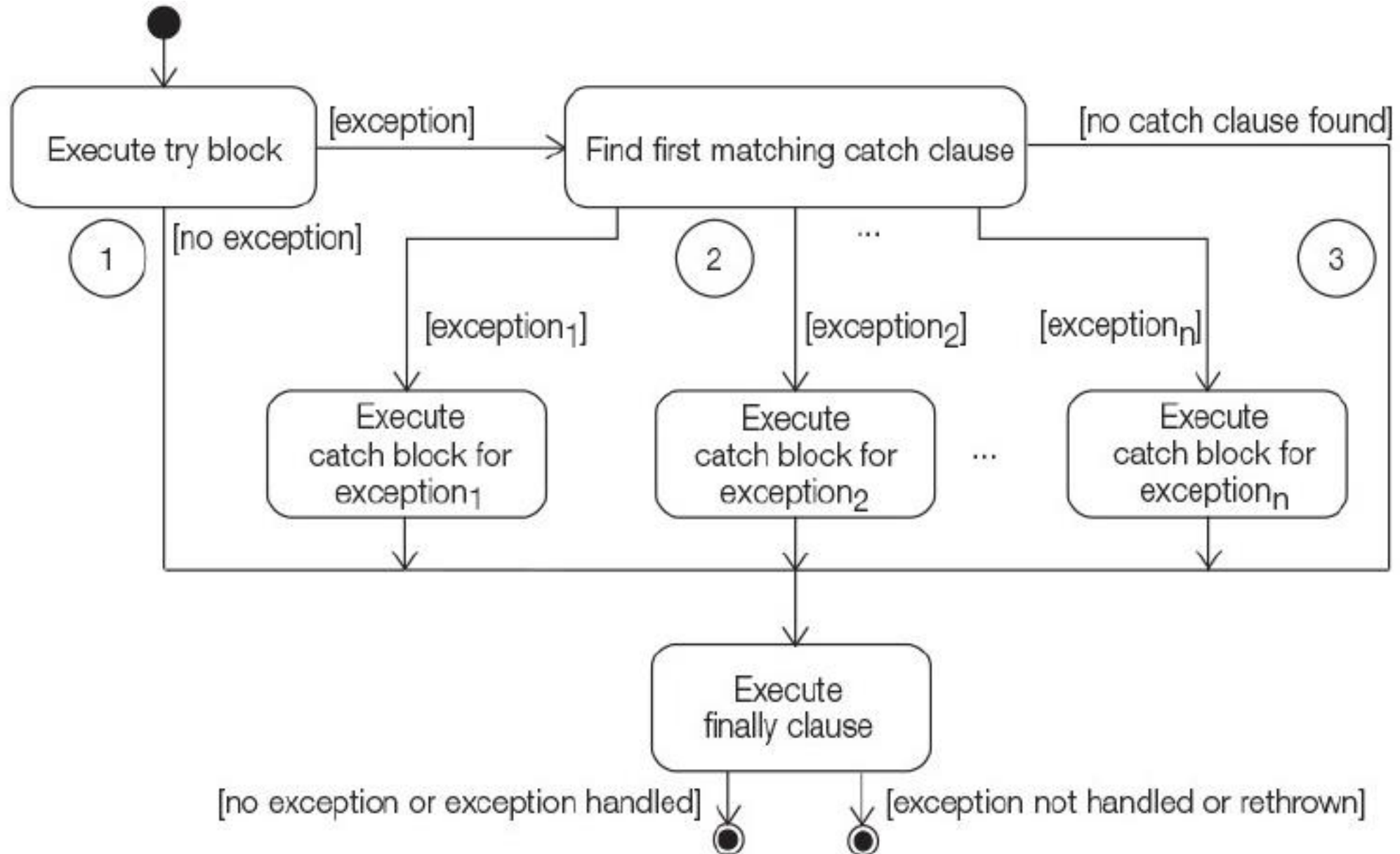
Handling Exceptions

- Differentiate among checked exceptions, unchecked exceptions, and Errors
- **Create a try-catch block and determine how exceptions alter normal program flow**
- Describe the advantages of Exception handling
- Create and invoke a method that throws an exception
- Recognize common exception classes (such as NullPointerException, ArithmeticException, ArrayIndexOutOfBoundsException, ClassCastException)



Обработка исключений

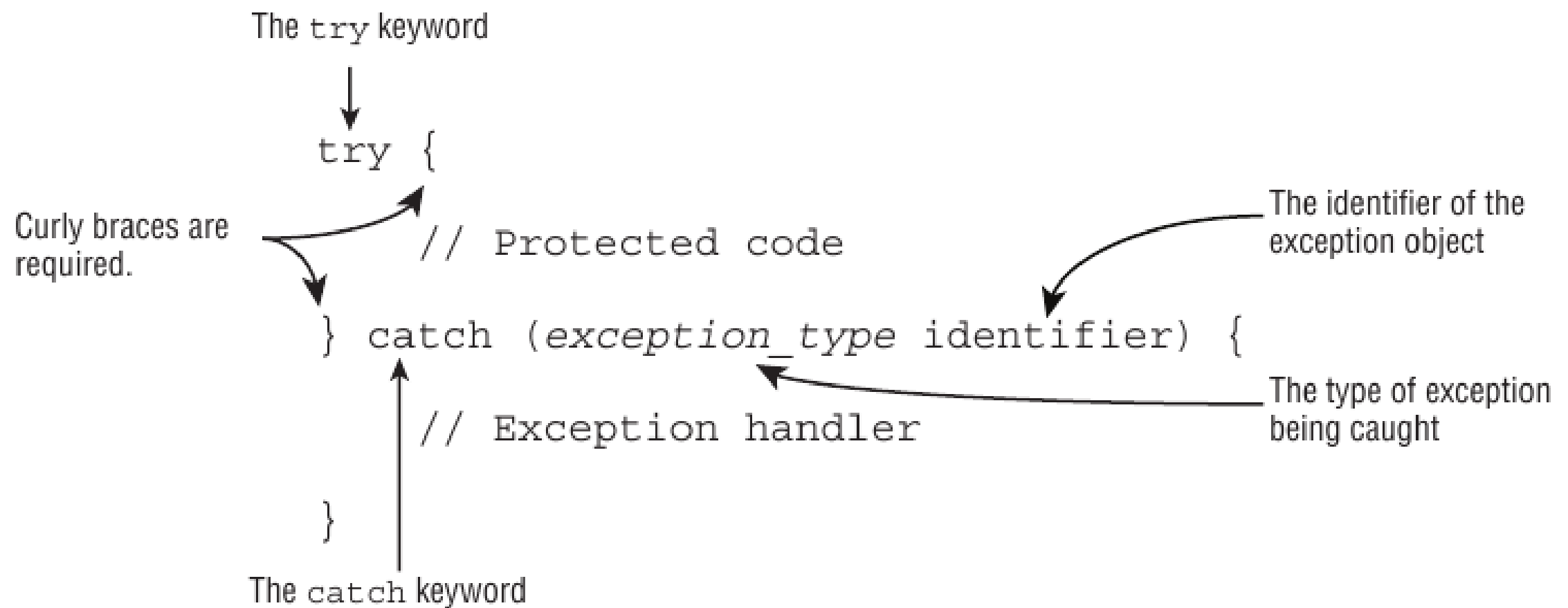
Блок-схема



Normal execution continues after try-catch-finally construct.

Execution aborted and exception propagated.

Синтаксис try



Простейший пример

```
void explore() {  
    try {  
        fall();  
        System.out.println("never get here");  
    } catch (RuntimeException e) {  
        getUp();  
    }  
    seeAnimals();  
}  
void fall() { throw new RuntimeException(); }
```

Простейшие ловушки

```
try // DOES NOT COMPILE
    fall();
catch (Exception e)
    System.out.println("get up");
```

```
try {
    fall();
} catch (Exception e) {
    System.out.println("get up");
}
```

```
try { // DOES NOT COMPILE
    fall();
}
```

catch блок

```
class AnimalsOutForAWalk extends RuntimeException { }  
class ExhibitClosed extends RuntimeException { }  
class ExhibitClosedForLunch extends ExhibitClosed { }
```

```
public void visitPorcupine() {  
    try {  
        seeAnimal();  
    } catch (AnimalsOutForAWalk e) { // first catch block  
        System.out.print("try back later");  
    } catch (ExhibitClosed e) { // second catch block  
        System.out.print("not today");  
    }  
}
```

Полиморфизм

```
public void visitMonkeys() {  
    try {  
        seeAnimal();  
    } catch (ExhibitClosedForLunch e) { // subclass exception  
        System.out.print("try back later");  
    } catch (ExhibitClosed e) { // superclass exception  
        System.out.print("not today");  
    }  
}
```

```
public void visitMonkeys() {  
    try {  
        seeAnimal();  
    } catch (ExhibitClosed e) {  
        System.out.print("not today");  
    } catch (ExhibitClosedForLunch e) { // DOES NOT COMPILE  
        System.out.print("try back later");  
    }  
}
```

Полиморфизм

```
public void visitSnakes() {  
    try {  
    } catch (IllegalArgumentException e) {  
    } catch (NumberFormatException e) { // DOES NOT COMPILE  
    }  
}
```

Область действия

```
public void visitManatees() {  
    try {  
    } catch (NumberFormatException e1) {  
        System.out.println(e1);  
    } catch (IllegalArgumentException e2) {  
        System.out.println(e1); // DOES NOT COMPILE  
    }  
}
```


Выбор catch

```
public class App {  
    public static void main(String[] args) {  
        try {  
            Throwable t = new Exception(); // ссылка типа Throwable указывает на объект типа  
Exception  
            throw t;  
        } 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");  
    }  
}  
  
>> catch Exception  
>> next statement
```

multi-catch блок

```
public static void main(String args[]) {  
    try {  
        System.out.println(Integer.parseInt(args[1]));  
    } catch (ArrayIndexOutOfBoundsException e) {  
        System.out.println("Missing or invalid input");  
    } catch (NumberFormatException e) {  
        System.out.println("Missing or invalid input");  
    }  
}
```

```
public static void main(String[] args) {  
    try {  
        System.out.println(Integer.parseInt(args[1]));  
    } catch (ArrayIndexOutOfBoundsException | NumberFormatException e) {  
        System.out.println("Missing or invalid input");  
    }  
}
```

Синтаксис multi-catch

```
try {  
    // Protected code  
} catch (Exception1 | Exception2 e) {  
    // Exception handler  
}
```

The diagram illustrates the multi-catch syntax with the following annotations:

- Catch either of these exceptions.**: An arrow points from this text to the pipe character (`|`) between `Exception1` and `Exception2` in the catch clause.
- Single identifier for all exception types**: An arrow points from this text to the identifier `e` in the catch clause.
- Required | between exception types**: An arrow points from this text to the pipe character (`|`) between `Exception1` and `Exception2` in the catch clause.
- // Protected code**: An arrow points from this text to the code block inside the `try` statement.
- // Exception handler**: An arrow points from this text to the code block inside the `catch` statement.

Одинокая переменная

```
catch(Exception1 e | Exception2 e | Exception3 e) // DOES NOT COMPILE
```

```
catch(Exception1 e1 | Exception2 e2 | Exception3 e3) // DOES NOT COMPILE
```

```
catch(Exception1 | Exception2 | Exception3 e)
```

Избыточность

```
try {  
    throw new IOException();  
} catch (FileNotFoundException | IOException p) {} // DOES NOT COMPILE
```

The exception `FileNotFoundException` is already caught by the alternative `IOException`

```
try {  
    throw new IOException();  
} catch (IOException e) { }
```

Пример

```
11: public void doesNotCompile() { // METHOD DOES NOT COMPILE
12:     try {
13:         mightThrow();
14:     } catch (FileNotFoundException | IllegalStateException e) {
15:     } catch (InputMismatchException e | MissingResourceException e) {
16:     } catch (FileNotFoundException | IllegalArgumentException e) {
17:     } catch (Exception e) {
18:     } catch (IOException e) {
19:     }
20: }
21: private void mightThrow() throws DateTimeParseException, IOException { }
```

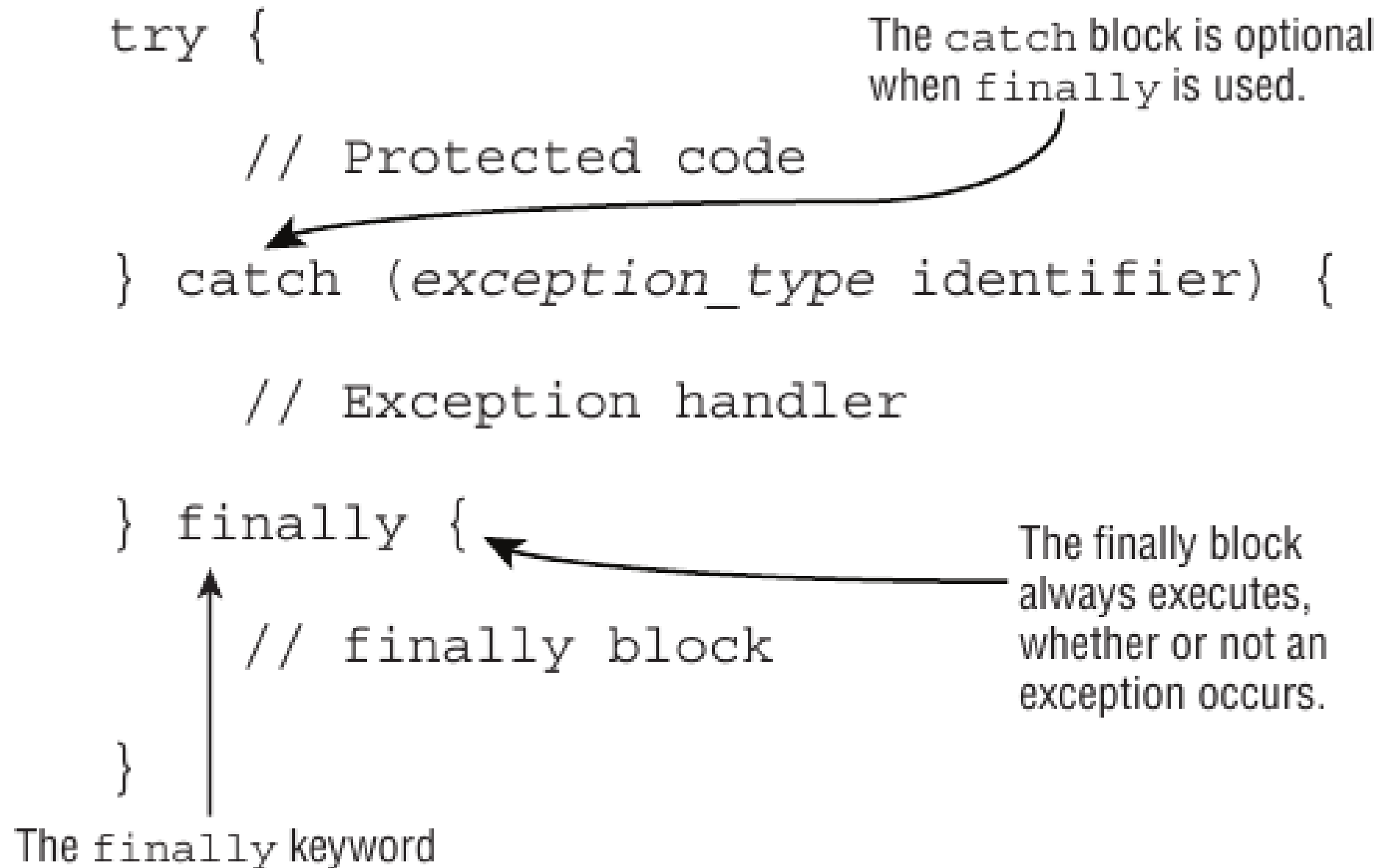
Синтаксис `finally`

```
try {  
    // Protected code  
} catch (exception_type identifier) {  
    // Exception handler  
} finally {  
    // finally block  
}
```

The `finally` keyword

The catch block is optional when `finally` is used.

The finally block always executes, whether or not an exception occurs.

The diagram illustrates the syntax of a try-finally block. It shows a code snippet with three main parts: a try block containing protected code, an optional catch block for exception handling, and a finally block that always executes. Annotations with arrows point to specific parts of the code: 'The catch block is optional when finally is used.' points to the catch block; 'The finally block always executes, whether or not an exception occurs.' points to the finally block; and 'The finally keyword' points to the 'finally' keyword in the finally block.

Простейший пример

```
void explore() {  
    try {  
        seeAnimals();  
        fall();  
    } catch (Exception e) {  
        getHugFromDaddy();  
    } finally {  
        seeMoreAnimals();  
    }  
    goHome();  
}
```


Ловушки

```
try { // DOES NOT COMPILE
    fall();
} finally {
    System.out.println("all better");
} catch (Exception e) {
    System.out.println("get up");
}
```

```
try { // DOES NOT COMPILE
    fall();
}
```

```
try {
    fall();
} finally {
    System.out.println("all better");
}
```

Что на выходе?

```
String s = "";  
try {  
    s += "t";  
} catch(Exception e) {  
    s += "c";  
} finally {  
    s += "f";  
}  
s += "a";  
System.out.print(s);
```

Что на выходе?

```
int goHome() {  
    try {  
        // Optionally throw an exception here  
        System.out.print("1");  
        return -1;  
    } catch (Exception e) {  
        System.out.print("2");  
        return -2;  
    } finally {  
        System.out.print("3");  
        return -3;  
    }  
}
```

Вето

```
public class Test {
    String testRef(){
        String str = "trying: ";
        try{ throw new NullPointerException(); }
        catch(NullPointerException npe){ return str + "caught";}
        finally{
            str += "finalized";           // не влияет на результат...
//            return str;                 // ...зато эта строка меняет его на
                                         // 'trying: finalized'
        }
    }

    int testPrim(){
        try{ throw new NullPointerException(); }
        catch(NullPointerException npe){ return 10;}
        finally { return 20; }           // примитивы тоже можно изменять
    }

    public static void main(String[] args) {
        Test t = new Test();
        System.out.println(t.testRef()); // печатает 'trying: caught'
        System.out.println(t.testPrim()); // печатает 20
    }
}
```

Всегда, но не до конца

```
} finally {  
    info.printDetails();  
    System.out.print("Exiting");  
    return "zoo";  
}
```

И всё же не всегда

```
try {  
    System.exit(0);  
} finally {  
    System.out.print("Never going to get here"); // Not printed  
}
```

Вложенные исключения

```
try {  
    throw new RuntimeException();  
} catch (RuntimeException e) {  
    throw new RuntimeException();  
} finally {  
    throw new Exception();  
}
```

Вложенные исключения

```
public class Test {  
    void run() {  
        try{  
            System.out.println("in try");  
            throw new NullPointerException();  
        }  
        catch(NullPointerException npe) {  
            System.out.println("caught NPE");  
            throw new RuntimeException();           // вызывает RTE  
        }  
        catch(RuntimeException rte) {             // не поймает вышеуказанное RTE  
            System.out.println("caught RTE");  
        }  
    }  
    public static void main(String[] args) {  
        Test t = new Test();  
        t.run();                                   // бросает RuntimeException  
    }  
}
```


ЛОВИМ ИСКЛЮЧЕНИЯ

```
public class App {  
    public static void main(String[] args) {  
        try {  
            System.err.print(" 0");  
            if (true) {throw new RuntimeException();}  
            System.err.print(" 1");  
        } catch (RuntimeException e) { // перехватили RuntimeException  
            System.err.print(" 2.1");  
            try {  
                System.err.print(" 2.2");  
                if (true) {throw new Error();} // и бросили новый Error  
                System.err.print(" 2.3");  
            } catch (Throwable t) { // перехватили Error  
                System.err.print(" 2.4");  
            }  
            System.err.print(" 2.5");  
        } catch (Error e) { // хотя есть catch по Error "ниже", но мы в него не попадаем  
            System.err.print(" 3");  
        }  
        System.err.println(" 4");  
    }  
}
```

```
>> 0 2.1 2.2 2.4 2.5 4
```

Что на выходе?

```
public String exceptions() {  
    String result = "";  
    String v = null;  
    try {  
        try {  
            result += "before";  
            v.length();  
            result += "after";  
        } catch (NullPointerException e) {  
            result += "catch";  
            throw new RuntimeException();  
        } finally {  
            result += "finally";  
            throw new Exception();  
        }  
    } catch (Exception e) {  
        result += "done";  
    }  
    return result;  
}
```

Завершающее действие

```
// open some resource
try {
    // use resource
} finally {
    // close resource
}
```

```
Lock lock = new ReentrantLock();
...
lock.lock();
try {
    // some code
} finally {
    lock.unlock();
}
```

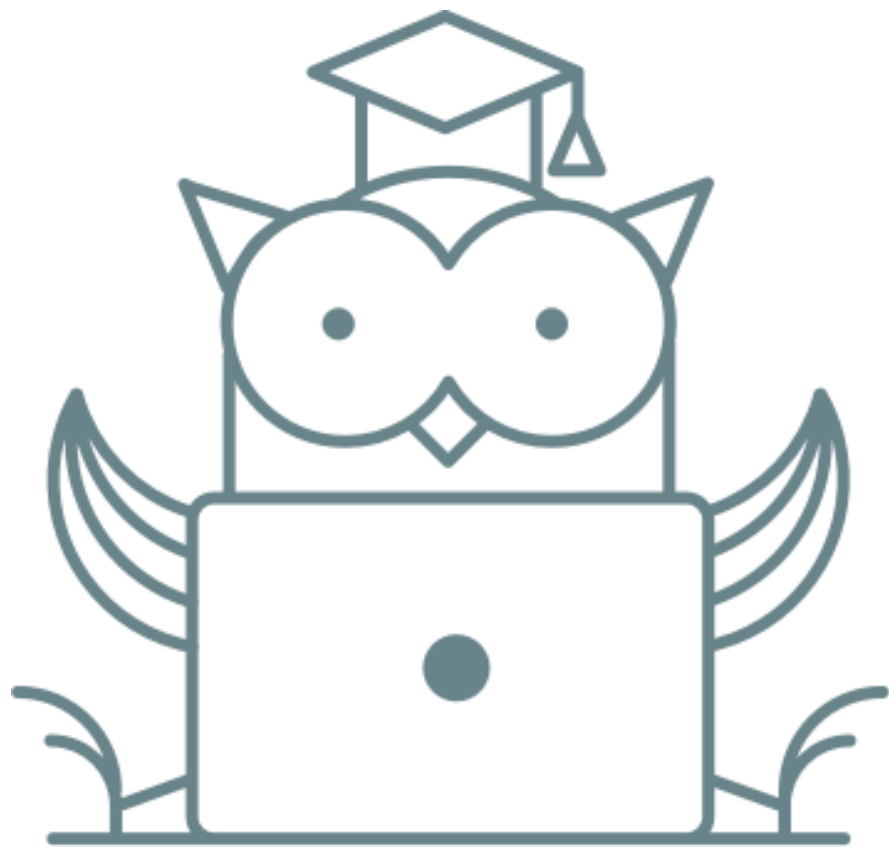
```
InputStream input = new FileInputStream("...");
try {
    // some code
} finally {
    input.close();
}
```

Было ли исключение?

```
public class App {  
    public static void main(String[] args) {  
        System.err.println(f());  
    }  
    public static int f() {  
        long rnd = System.currentTimeMillis();  
        boolean finished = false;  
        try {  
            if (rnd % 3 == 0) {  
                throw new Error();  
            } else if (rnd % 3 == 1) {  
                throw new RuntimeException();  
            } else {  
                // nothing  
            }  
            finished = true;  
        } finally {  
            if (finished) {  
                // не было исключений  
            } else {  
                // было исключение, но какое?  
            }  
        }  
    }  
}
```



Вопросы?



Поток исполнения

try + finally

```
public class App {  
    public static void main(String[] args) {  
        try {  
            System.err.println("try");  
        } finally {  
            System.err.println("finally");  
        }  
    }  
}  
  
>> try  
>> finally
```

try + finally

```
public class App {  
    public static void main(String[] args) {  
        try {  
            throw new RuntimeException();  
        } finally {  
            System.err.println("finally");  
        }  
    }  
}  
  
>> finally  
>> Exception in thread "main" java.lang.RuntimeException
```


try + finally

```
public class App {  
    public static void main(String[] args) {  
        try {  
            Runtime.getRuntime().exit(42);  
        } finally {  
            System.err.println("finally");  
        }  
    }  
}
```

```
>> Process finished with exit code 42
```

try + finally

```
public class App {  
    public static void main(String[] args) {  
        try {  
            Runtime.getRuntime().halt(42);  
        } finally {  
            System.err.println("finally");  
        }  
    }  
}
```

>> Process finished with `exit` code 42

try + finally

```
public class App {  
    public static void main(String[] args) {  
        try {  
            System.err.println("try");  
            if (true) {throw new RuntimeException();}  
        } finally {  
            System.err.println("finally");  
        }  
        System.err.println("more");  
    }  
}  
  
>> try  
>> finally  
>> Exception in thread "main" java.lang.RuntimeException
```

try + finally

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

>> COMPILER ERROR: Unreachable statement

try + finally

```
public class App {  
    public static void main(String[] args) {  
        try {  
            System.err.println("try");  
            if (true) {return;}  
        } finally {  
            System.err.println("finally");  
        }  
        System.err.println("more");  
    }  
}  
  
>> try  
>> finally
```

try + finally

```
public class App {  
    public static void main(String[] args) {  
        System.err.println(f());  
    }  
    public static int f() {  
        try {  
            return 0;  
        } finally {  
            return 1;  
        }  
    }  
}  
  
>> 1
```

try + finally

```
public class App {  
    public static void main(String[] args) {  
        System.err.println(f());  
    }  
    public static int f() {  
        try {  
            throw new RuntimeException();  
        } finally {  
            return 1;  
        }  
    }  
}  
  
>> 1
```

try + finally

```
public class App {  
    public static void main(String[] args) {  
        System.err.println(f());  
    }  
    public static int f() {  
        try {  
            return 0;  
        } finally {  
            throw new RuntimeException();  
        }  
    }  
}  
  
>> Exception in thread "main" java.lang.RuntimeException
```


try + finally

```
public class App {  
    public static void main(String[] args) {  
        System.err.println(f());  
    }  
    public static int f() {  
        try {  
            throw new Error();  
        } finally {  
            throw new RuntimeException();  
        }  
    }  
}  
  
>> Exception in thread "main" java.lang.RuntimeException
```

Упражнение

```
public class RQ6A19 {  
    public static void main(String[] args) throws InterruptedException {  
        try {  
            throwIt();  
            System.out.println("1");  
        } finally {  
            System.out.println("2");  
        }  
        System.out.println("3");  
    }  
  
    // InterruptedException is a direct subclass of Exception.  
    static void throwIt() throws InterruptedException {  
        throw new InterruptedException("Time to go home.");  
    }  
}
```

Select the one correct answer.

- (a) The program will print 2 and throw `InterruptedException`.
- (b) The program will print 1 and 2, in that order.
- (c) The program will print 1, 2, and 3, in that order.
- (d) The program will print 2 and 3, in that order.
- (e) The program will print 3 and 2, in that order.
- (f) The program will print 1 and 3, in that order.



Ответ: A

Упражнение

```
public class Exceptions {  
    public static void main(String[] args) {  
        try {  
            if (args.length == 0) return;  
            System.out.println(args[0]);  
        } finally {  
            System.out.println("The end");  
        }  
    }  
}
```

Select the two correct answers.

- (a) If run with no arguments, the program will produce no output.
- (b) If run with no arguments, the program will print The end.
- (c) The program will throw an `ArrayIndexOutOfBoundsException`.
- (d) If run with one argument, the program will simply print the given argument.
- (e) If run with one argument, the program will print the given argument followed by "The end".



Ответ: ВЕ

try + catch + finally

```
public class App {  
    public static void main(String[] args) {  
        try {  
            System.err.print(" 0");  
            // nothing  
            System.err.print(" 1");  
        } catch (Error e) {  
            System.err.print(" 2");  
        } finally {  
            System.err.print(" 3");  
        }  
        System.err.print(" 4");  
    }  
}
```

```
>> 0 1 3 4
```

try + catch + finally

```
public class App {  
    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");  
    }  
}
```

```
>> 0 2 3 4
```

try + catch + finally

```
public class App {  
    public static void main(String[] args) {  
        try {  
            System.err.print(" 0");  
            if (true) {throw new RuntimeException();}  
            System.err.print(" 1");  
        } catch(Error e) {  
            System.err.print(" 2");  
        } finally {  
            System.err.print(" 3");  
        }  
        System.err.print(" 4");  
    }  
}  
  
>> 0 3  
>> RUNTIME ERROR: Exception in thread "main" java.lang.RuntimeException
```


Вложенные TCF

```
public class App {  
    public static void main(String[] args) {  
        try {  
            System.err.print(" 0");  
            try {  
                System.err.print(" 1");  
                // НИЧЕГО  
                System.err.print(" 2");  
            } catch (RuntimeException e) {  
                System.err.print(" 3"); // НЕ заходим - нет исключения  
            } finally {  
                System.err.print(" 4"); // заходим всегда  
            }  
            System.err.print(" 5"); // заходим - выполнение в норме  
        } catch (Exception e) {  
            System.err.print(" 6"); // НЕ заходим - нет исключения  
        } finally {  
            System.err.print(" 7"); // заходим всегда  
        }  
        System.err.print(" 8"); // заходим - выполнение в норме  
    }  
}
```

>> 0 1 2 4 5 7 8

Вложенные TCF

```
public class App {  
    public static void main(String[] args) {  
        try {  
            System.err.print(" 0");  
            try {  
                System.err.print(" 1");  
                if (true) {throw new RuntimeException();}  
                System.err.print(" 2");  
            } catch (RuntimeException e) {  
                System.err.print(" 3"); // ЗАХОДИМ - есть исключение  
            } finally {  
                System.err.print(" 4"); // заходим всегда  
            }  
            System.err.print(" 5"); // заходим - выполнение УЖЕ в норме  
        } catch (Exception e) {  
            System.err.print(" 6"); // не заходим - нет исключения, УЖЕ перехвачено  
        } finally {  
            System.err.print(" 7"); // заходим всегда  
        }  
        System.err.print(" 8"); // заходим - выполнение УЖЕ в норме  
    }  
}
```

>> 0 1 3 4 5 7 8

Вложенные TCF

```
public class App {
    public static void main(String[] args) {
        try {
            System.err.print(" 0");
            try {
                System.err.print(" 1");
                if (true) {throw new Exception();}
                System.err.print(" 2");
            } catch (RuntimeException e) {
                System.err.print(" 3"); // НЕ заходим - есть исключение, но НЕПОДХОДЯЩЕГО ТИ
            }

            } finally {
                System.err.print(" 4"); // заходим всегда
            }
            System.err.print(" 5"); // не заходим - выполнение НЕ в норме
        } catch (Exception e) {
            System.err.print(" 6"); // ЗАХОДИМ - есть подходящее исключение
        } finally {
            System.err.print(" 7"); // заходим всегда
        }
        System.err.print(" 8"); // заходим - выполнение УЖЕ в норме
    }
}
```

ПА

>> 0 1 4 6 7 8

Вложенные TCF

```
public class App {  
    public static void main(String[] args) {  
        try {  
            System.err.print(" 0");  
            try {  
                System.err.print(" 1");  
                if (true) {throw new Error();}  
                System.err.print(" 2");  
            } catch (RuntimeException e) {  
                System.err.print(" 3"); // НЕ заходим - есть исключение, но НЕПОДХОДЯЩЕГО ТИ  
  
                } finally {  
                    System.err.print(" 4"); // заходим всегда  
                }  
                System.err.print(" 5"); // НЕ заходим - выполнение НЕ в норме  
            } catch (Exception e) {  
                System.err.print(" 6"); // не заходим - есть исключение, но НЕПОДХОДЯЩЕГО ТИ  
  
                } finally {  
                    System.err.print(" 7"); // заходим всегда  
                }  
                System.err.print(" 8"); // не заходим - выполнение НЕ в норме  
            }  
        }  
    }  
}
```

>> 0 1 4 7

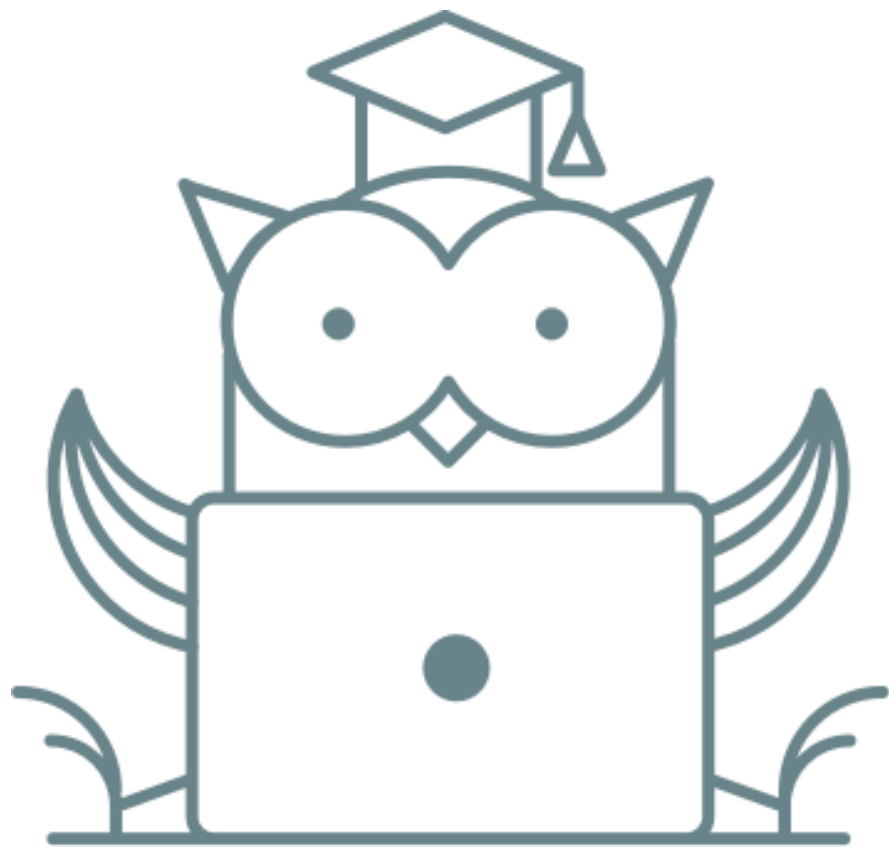
>> RUNTIME EXCEPTION: Exception in thread "main" java.lang.Error

Упражнение

```
public class Test{
    public double test(double d){
        if(d > 0.5) return 1;
        try{
            if(d < 0.1) throw new Exception("Too small!"); // line t1
        }
        catch(Exception e) {
            return 666; // line t2
        }
        finally {
            return 42; // line t3
        }
    }
    public static void main(String args[]){
        System.out.println(new Test().test(Math.random()));
    }
}
```

Which one is true?

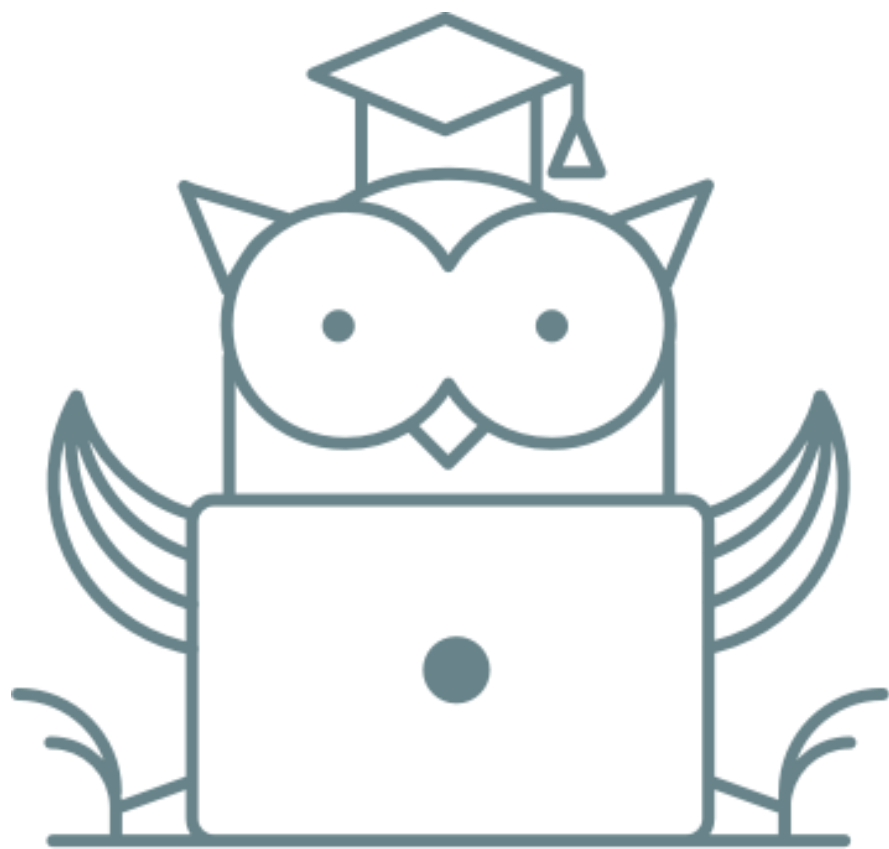
- A. The code always prints 42
- B. The code will fail compilation if t1 is commented out
- C. The code compiles if both t1 and t2 are commented out
- D. None of the above



Ответ: С



Вопросы?



**Код
недоступен**

Антипаттерн

Если вслед за `catch` или `finally` идут какие-то операции, они (операции) будут исполнены, ЗА ИСКЛЮЧЕНИЕМ ситуации, когда:

- 1) ВСЕ `try`- и `catch`-блоки содержат `return` или бросают какое-то исключение;

или

- 2) `finally` содержит `return` или бросает какое-то исключение.

В этом случае упомянутые операции становятся недоступными, вызывая комперр.

То же короче

Операции, стоящие после обработки исключений, оказываются недоступными, если имеется безусловный `return` (или выброс исключения – `throw`) в `finally` или внутри как `try`, так и `catch`.

Пример

```
class Test {
    public float parseFloat(String s){
        float f = 0.0f;
        try{
            f = Float.valueOf(s).floatValue();
            return f ;
        }
        catch(NumberFormatException nfe){
            System.out.println("Invalid input " + s);
            f = Float.NaN ;
            return f;
        }
        finally {
            System.out.println("finally " + f);
        }
        return f ;
    }
}

// ----- happy path -----
// ----- unhappy path -----

public static void main(String[] args) {
    Test t = new Test();
    t.parseFloat("1.1");
    t.parseFloat("one-point-one");
}
```

Упражнение

```
public class ManyHappyReturns {
    static int run(){
        int a = Math.random() > 0.5 ? 1 : 0;
        try {
            return 1/a ;                // line X
        }
        catch(ArithmeticException ae){
            return 666;                // line XX
        }
        finally {
            return 42;                // line XXX
        }
        return 123;                // line XXXX
    }
    public static void main(String[] args) {
        System.out.println(run());
    }
}
```

Which four are true?

- A. If lines X and XX are commented out, the code prints 42
- B. If line XXX is commented out, the code prints either 1 or 666
- C. If lines X and XXX are commented out, the code prints 123
- D. If line XX is commented out, the code prints 123
- E. If lines XX and XXX are commented out, the code prints either 1 or 123
- F. If line XXXX is commented out, the code prints 42
- G. If lines XXX and XXXX are commented out, the code prints either 1 or 666



Ответ: CEFG

Упражнение

```
1 import java.io.IOException;
2
3 public class StayAlert {
4     public static void main(String args[]) {
5         IOException ioe = null;
6         try {
7             throw null;
8         }
9         catch (NullPointerException npe) { System.out.print("Caught NPE "); }
10        try {
11            throw ioe;
12        }
13        catch (IOException ioe) { System.out.print("+ Caught IOE"); }
14    }
15 }
```

Какие два утверждения верны?

- A. try-catch на строках 10 – 13 необходим
- B. try-catch на строках 6 – 9 избыточен
- C. Код печатает Caught NPE + Caught IOE
- D. На этапе исполнения возбуждается **RuntimeException**
- E. Компиляция не проходит



Ответ: АЕ



Вопросы?

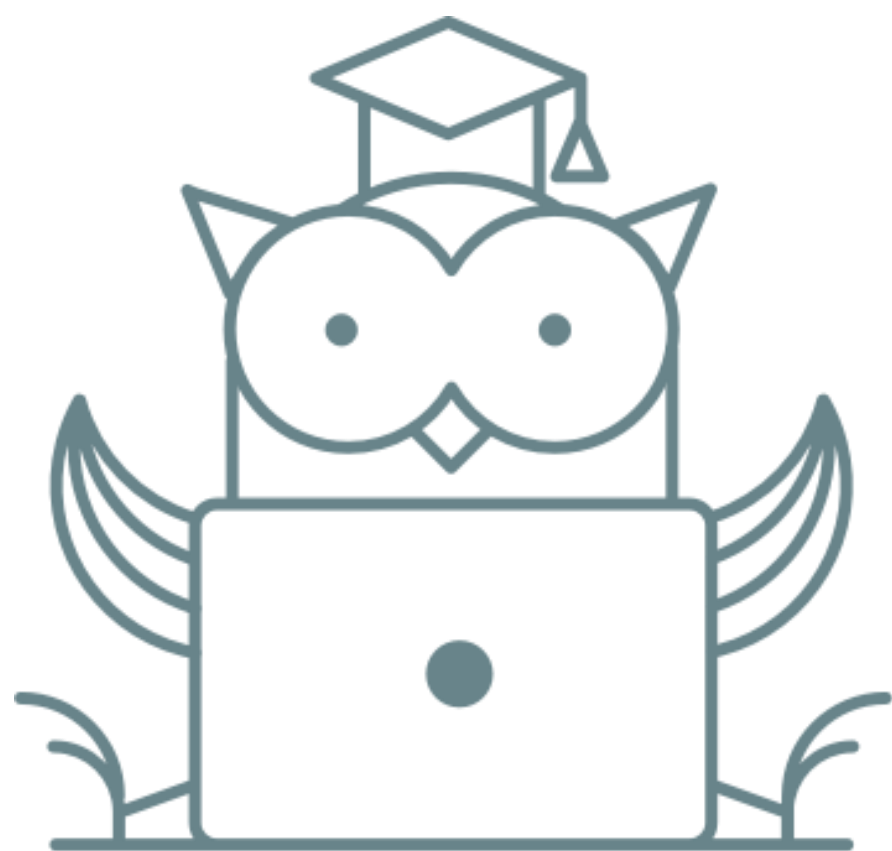
Домашнее задание

Тест



Пожалуйста, пройдите опрос

<https://otus.ru/polls/17838/>



**Спасибо
за внимание!**

**Плавайте в потоке
исключений смело!**