

Programozási nyelvek – Java

Hibák és kivételek



Kozsik Tamás

ELTE Eötvös Loránd Tudományegyetem

1 Hiba detektálása és jelzése

- assert utasítás
- Dokumentációs megjegyzés

2 Kivételek

- Kivételkezelés

3 Szövegfájlok

Hibajelzés kivétel kiváltásával

```
public class Time {  
    private int hour, minute;    // 0 <= hour < 24, 0 <= minute < 60  
    public Time( int hour, int minute ){ ... }  
    public int getHour(){ return hour; }  
    public int getMinute(){ return minute; }  
    public void setHour( int hour ){  
        if( 0 <= hour && hour <= 23 ){  
            this.hour = hour;  
        } else {  
            throw new IllegalArgumentException("Invalid hour!");  
        }  
    }  
    public void setMinute( int minute ){ ... }  
    public void aMinutePassed(){ ... }  
}
```



Az assert utasítás

```
public class Time {  
    private int hour, minute;    // 0 <= hour < 24, 0 <= minute < 60  
    public Time( int hour, int minute ){ ... }  
    public int getHour(){ return hour; }  
    public int getMinute(){ return minute; }  
  
    // may throw AssertionError  
    public void setHour( int hour ){  
        assert 0 <= hour && hour <= 23 ;  
        this.hour = hour;  
    }  
  
    public void setMinute( int minute ){ ... }  
    public void aMinutePassed(){ ... }  
}
```



Az assert utasítás

TestTime.java

```
Time time = new Time(6,30);  
time.setHour(30);
```

Futtatás

```
$ java TestTime  
$ java -enableassertions TestTime  
Exception in thread "main" java.lang.AssertionError  
    at Time.setHour(Time.java:7)  
    at TestTime.main(TestTime.java:5)  
$
```



Dokumentációs megjegyzés

```
/** May throw AssertionError. */  
public void setHour( int hour ){  
    assert 0 <= hour && hour <= 23 ;  
    this.hour = hour;  
}
```



Dokumentált potenciálisan hibás használat

```
/**  
    Blindly sets the hour property to the given value.  
    Use it with care: only pass {@code hour} satisfying  
    {@code 0 <= hour && hour <= 23}.  
*/  
public void setHour( int hour ){  
    this.hour = hour;  
}
```



javadoc Time.java

PACKAGE **CLASS** TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SEARCH:

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Constructor Summary

Constructors

Constructor	Description
<code>Time()</code>	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
int	<code>getHour()</code>	
int	<code>getMinute()</code>	
void	<code>oneMinutePassed()</code>	
void	<code>setHour(int hour)</code>	Blindly sets the hour property to the given value.



javadoc Time.java

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)SEARCH: [SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

getHour

```
public int getHour()
```

getMinute

```
public int getMinute()
```

setHour

```
public void setHour(int hour)
```

Blindly sets the hour property to the given value. Use it with care: only pass hour satisfying $0 \leq \text{hour} \ \&\& \ \text{hour} \leq 23$.



Szokásos (túl bőbeszédű) dokumentációs megjegyzés

```
/**
 * Sets the hour property. Only pass an {@code hour}
 * satisfying {@code 0 <= hour && hour <= 23}.
 * @param hour The value to be set.
 * @throws IllegalArgumentException
 *     If the supplied value is not between 0 and 23,
 *     inclusively.
 */
public void setHour( int hour ){
    if( 0 <= hour && hour <= 23 ){
        this.hour = hour;
    } else {
        throw new IllegalArgumentException("Invalid hour!");
    }
}
```



javadoc Time.java

setHour

```
public void setHour(int hour)
```

Sets the hour property. Only pass an hour satisfying $0 \leq \text{hour} \leq 23$.

Parameters:

hour - The value to be set.

Throws:

`java.lang.IllegalArgumentException` - If the supplied value is not between 0 and 23, inclusively.



Szintaxiskiemelés

```
/**
 * Sets the hour property. Only pass an {@code hour}
 * satisfying {@code 0 <= hour && hour <= 23}.
 * @param hour The value to be set.
 * @throws IllegalArgumentException
 *     If the supplied value is not between 0 and 23,
 *     inclusively.
 */
public void setHour( int hour ){
    if( 0 <= hour && hour <= 23 ){
        this.hour = hour;
    } else {
        throw new IllegalArgumentException("Invalid hour!");
    }
}
```

21,1



Opciók hibák jelzésére

Jó megoldások

- `IllegalArgumentException`: modul határán
- `assert`: modul belsejében
- Dokumentációs megjegyzés

Rossz megoldások

- Csendben elszabotálni a műveletet
- Elsumákolni az ellenőrzéseket



- 1 Hiba detektálása és jelzése
 - assert utasítás
 - Dokumentációs megjegyzés
- 2 Kivételek
 - Kivételkezelés
- 3 Szövegfájlok

Ellenőrzött kivételek

checked exceptions

```
public Time readTime( String fname ) throws java.io.IOException {  
    ...  
}
```

- A programszövegben jelölni kell a terjedését
- A fordítóprogram ellenőrzi a konzisztenciát
- Ilyen: `java.sql.SQLException`, `java.security.KeyException`
- Nem ilyen: `NullPointerException`, `ArrayIndexOutOfBoundsException`



Nem ellenőrzött kivételek

unchecked exception

- Pl. NullPointerException, ArrayIndexOutOfBoundsException
- Dinamikus szemantikai hiba
- „Bárhol” ’ keletkezhet



Terjedés követése: fordítási hiba

```
import java.io.IOException;
class TestTime {
    public Time readTime( String fname ) throws IOException {
        ... new java.io.FileReader(fname) ...
    }

    public static void main( String[] args ){
        TestTime tt = new TestTime();
        Time wakeUp = tt.readTime("wakeup.txt");
        wakeUp.aMinutePassed();
    }
}
```



Terjedés követése: fordítási hiba javítva

```
import java.io.IOException;
class TestTime {
    public Time readTime( String fname ) throws IOException {
        ... new java.io.FileReader(fname) ...
    }

    public static void main( String[] args ) throws IOException {
        TestTime tt = new TestTime();
        Time wakeUp = tt.readTime("wakeup.txt");
        wakeUp.aMinutePassed();
    }
}
```



Kivételkezelés

```
import java.io.IOException;

class TestTime {
    public Time readTime( String fname ) throws IOException {
        ... new java.io.FileReader(fname) ...
    }
    public static void main( String[] args ){
        TestTime tt = new TestTime();
        try {
            Time wakeUp = tt.readTime("wakeup.txt");
            wakeUp.aMinutePassed();
        } catch( IOException e ){
            System.err.println("Could not read wake-up time.");
        }
    }
}
```



A program tovább futhat a probléma ellenére

```
public class Receptionist {  
    ...  
    public Time[] readWakeupTimes( String[] fnames ){  
        Time[] times = new Time[fnames.length];  
        for( int i = 0; i < fnames.length; ++i ){  
            try {  
                times[i] = readTime(fnames[i]);  
            } catch( java.io.IOException e ){  
                times[i] = null;    // no-op  
                System.err.println("Could not read " + fnames[i]);  
            }  
        }  
        return times; // maybe sort times before returning?  
    }  
}
```



A try-catch utasítás

```
<try-catch-statement> ::= try <block-statement>  
                           <catch-list>  
                           <optional-finally-part>
```

```
<catch-list> ::= ""  
               | <catch-part> <catch-list>
```

```
<catch-part> ::= catch (<exceptions> <identifier>)  
                    <block-statement>
```

```
<exceptions> ::= <identifier>  
                | <identifier> | <exceptions>
```

```
<optional-finally-part> ::= ""  
                          | finally <block-statement>
```



Több catch-ág

```
public static Time parse( String str ){
    String errorMessage;
    try {    String[] parts = str.split(":");
            int hour = Integer.parseInt(parts[0]);
            int minute = Integer.parseInt(parts[1]);
            return new Time(hour,minute);
        } catch( NullPointerException e ){
            errorMessage = "Null parameter is not allowed!";
        } catch( ArrayIndexOutOfBoundsException e ){
            errorMessage = "String must contain \":\"!";
        } catch( NumberFormatException e ){
            errorMessage = "String must contain two numbers!";
        }
    throw new IllegalArgumentException(errorMessage);
}
```



Egy catch-ágban több kivétel

```
public static Time parse( String str ){
    try {
        String[] parts = str.split(":");
        int hour = Integer.parseInt(parts[0]);
        int minute = Integer.parseInt(parts[1]);
        return new Time(hour,minute);
    } catch( NullPointerException
           | ArrayIndexOutOfBoundsException
           | NumberFormatException e ){
        throw new IllegalArgumentException("Can't parse time!");
    }
}
```



A try-finally utasítás

```
public static Time readTime( String fname ) throws IOException {  
    BufferedReader in = new BufferedReader(new FileReader(fname));  
    Time time;  
    try {  
        String line = in.readLine();  
        time = parse(line);  
    } finally {  
        in.close();  
    }  
    return time;  
}
```



A finally mindenképp vezérlést kap!

```
public static Time readTime( String fname ) throws IOException {  
    BufferedReader in = new BufferedReader(new FileReader(fname));  
    try {  
        String line = in.readLine();  
        return parse(line);  
    } finally {  
        in.close();  
    }  
}
```



A try-catch-finally utasítás

```
public static Time readTime( String fname ) throws IOException {  
    BufferedReader in = new BufferedReader(new FileReader(fname));  
    try {  
        String line = in.readLine();  
        return parse(line);  
    } catch ( IllegalArgumentException e ){  
        System.err.println(e);  
        System.err.println("Using default value!");  
        return new Time(0,0);  
    } finally {  
        in.close();  
    }  
}
```



A try-utasítások egymásba ágyazhatók

```
public static Time readTimeOrUseDefault( String fn ){
    try {
        BufferedReader in = new BufferedReader(new FileReader(fn));
        try {
            String line = in.readLine();
            return parse(line);
        } finally {
            in.close();
        }
    } catch( IOException | IllegalArgumentException e ){
        System.err.println(e);
        System.err.println("Using default value!");
        return new Time(0,0);
    }
}
```



Lényegében ekvivalensek

try-finally

```
BufferedReader in = ... ;  
try {  
    String line = in.readLine();  
    return parse(line);  
} finally {  
    in.close();  
}
```

try-with-resources

```
try(  
    BufferedReader in = ...  
) {  
    String line = in.readLine();  
    return parse(line);  
}
```



A *try-with-resources* utasítás

```
public static Time readTimeOrUseDefault( String fn ){
    try {
        try(
            BufferedReader in = new BufferedReader(new FileReader(fn))
        ){
            String line = in.readLine();
            return parse(line);
        }
    } catch( IOException | IllegalArgumentException e ){
        System.err.println(e);
        System.err.println("Using default value!");
        return new Time(0,0);
    }
}
```



A *try-with-resources* utasítással még egyszerűbben

```
public static Time readTimeOrUseDefault( String fn ){  
    try(  
        BufferedReader in = new BufferedReader(new FileReader(fn))  
    ){  
        String line = in.readLine();  
        return parse(line);  
    } catch( IOException | IllegalArgumentException e ){  
        System.err.println(e);  
        System.err.println("Using default value!");  
        return new Time(0,0);  
    }  
}
```



Több erőforrás használata

```
static void copy( String in, String out ) throws IOException {  
    try (  
        FileInputStream infile = new FileInputStream(in);  
        FileOutputStream outfile = new FileOutputStream(out)  
    ){  
        int b;  
        while( (b = infile.read()) != -1 ){    // idióma!  
            outfile.write(b);  
        }  
    }  
}
```



- 1 Hiba detektálása és jelzése
 - assert utasítás
 - Dokumentációs megjegyzés
- 2 Kivételek
 - Kivételkezelés
- 3 Szövegfájlok

Karakterkódolási szabványok

character encodings

- Bacon's cipher, 1605 (Francis Bacon)
- Baudot-code, 1874
- BCDIC, 1928 (Binary Coded Decimal Interchange Code)
- EBCDIC, 1963 (Extended ...)
- ASCII, 1963 (American Standard Code for Information Interchange)
- ISO/IEC 8859 (Latin-1, Latin-2,...)
- Windows 1250 (Cp1250)
- Unicode (UTF-8, UTF-16, UTF-32)

lásd: `iconv` (Unix/Linux)



Szövegfájlok írása

```
import java.io.*;
```

Kiírás a platform alapértelmezett kódolásával

```
try( FileWriter out = new FileWriter(fname) ){  
    out.write("árvíztűrő ütvefűrógép", 0, 21);  
}
```



Szövegfájlok írása

```
import java.io.*;
```

Kiírás a platform alapértelmezett kódolásával

```
try( FileWriter out = new FileWriter(fname) ){  
    out.write("árvíztűrő ütvefúrógép", 0, 21);  
}
```

Karakterkódolás explicit megadása

```
try( OutputStreamWriter out = new OutputStreamWriter(  
    new FileOutputStream(fname),  
    "Cp1250"  
    ) ){  
    out.write("árvíztűrő ütvefúrógép", 0, 21);  
}
```

Szövegfájlok olvasása soronként

```
import java.io.*;
```

Karakterkódolás explicit megadása

```
try(  
    BufferedReader in = new BufferedReader(  
        new InputStreamReader(  
            new FileInputStream(fname),  
            "Cp1250"  
        )  
    )  
) {  
    String line;  
    while( (line=in.readLine()) != null ) { ... }  
}
```

