

Programare Orientată pe Obiecte

Tratarea Excepțiilor în Java

Dr. Petru Florin Mihancea

V20180924

Programare Orientată pe Obiecte

Tratarea Excepțiilor în Java

Dr. Petru Florin Mihancea

V20180924

Programare Orientată pe Obiecte

Tratarea Excepțiilor în Java

Dr. Petru Florin Mihancea

V20180924

Programare Orientată pe Obiecte

Tratarea Excepțiilor în Java

Dr. Petru Florin Mihancea

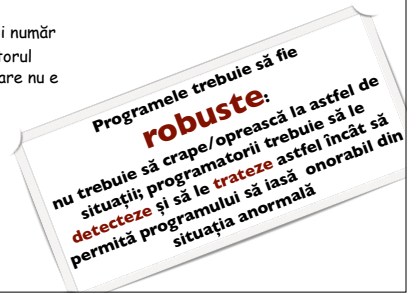
V20180924

Situație Excepțională

O **excepție** este un eveniment ce apare la execuția unui program și care disturbă modul uzual în care programul se execută

Exemplu

Programul cere introducerea unui număr întreg de la tastatură iar utilizatorul furnizează un șir de caractere care nu e număr întreg



Situație Excepțională (II)

Detecția situației anormale și tratarea ei

- uzual în locuri diferite de program, în metode diferite
- în locul detecției problemei nu avem uzual suficiente informații pentru a elimina problema (dacă am avea, am rezova-o direct)
- în locul detecției nu mai putem continua procesarea

Exemplu (continuare)

`Integer.parseInt(String)` detectează dacă argumentul nu e număr întreg
... dar tratarea situației este/ poate fi făcută acolo unde "numărul" e citit

prin urmare e necesară o "comunicare" între
cele două puncte din program, cel puțin pt. a anunța
apariția situației anormale

A

Fanioane și valori de return speciale

Dr. Petru Florin Mihancea

setTime poate detecta
încercarea de a seta o oră
invalidă

Setarea pe 0 NU e o soluție prea generală!
Cine ar putea ști mai exact ce
trebuie să facă dacă ora e
invalidă ?

Apelantul lui setTime,
sau apelantul apelantului lui setTime, etc.

Exemplu I

```
class Clock {  
    private int hour, minute, second;  
  
    public Clock() {  
        hour = minute = second = 0;  
    }  
  
    public void setTime(int h, int m, int s) {  
        hour = (h >= 0) && (h < 24) ? h : 0;  
        minute = (m >= 0) && (m < 60) ? m : 0;  
        second = (s >= 0) && (s < 60) ? s : 0;  
    }  
  
    public String toString() {  
        return "Current time " + hour + ":" +  
            minute + ":" + second;  
    }  
}
```

Dar cum îl anunțăm
că ceva nu e OK ?

Exemplu I

```
public void doSomething1() {  
    Clock cf;  
    int a, b, c;  
    //... cod normal ce obtine a, b, c, cf  
    cf.setTime(a, b, c);  
    //... cod normal  
}
```

Apelantul **NU** poate fi obligat să facă verificări

```
class Clock {  
    private int hour, minute, second;  
  
    public Clock() {  
        hour = minute = second = 0;  
    }  
  
    public int setTime(int h, int m, int s) {  
        if((h < 0) || (h > 23)) return -1;  
        if((m < 0) || (m > 59)) return -2;  
        if((s < 0) || (s > 59)) return -3;  
        hour = h;  
        minute = m;  
        second = s;  
        return 0;  
    }  
  
    public String toString() {  
        return "Current time " + hour + ":" +  
            minute + ":" + second;  
    }  
}
```

Exemplu I

```
public void doSomething2() {  
    Clock cf;  
    int a, b, c;  
    //... cod normal ce obtine a, b, c, cf  
    int res = cf.setTime(a, b, c);  
    if(res != 0) {  
        //... cod tratare situatie anormala  
    } else {  
        //... cod normal  
    }  
}
```

Codul pentru cazul uzual se
mixează cu codul tratării
situațiilor excepționale
(deci va fi mai greu de citit / înțeles)

```
class Clock {  
    private int hour, minute, second;  
  
    public Clock() {  
        hour = minute = second = 0;  
    }  
  
    public int setTime(int h, int m, int s) {  
        if((h < 0) || (h > 23)) return -1;  
        if((m < 0) || (m > 59)) return -2;  
        if((s < 0) || (s > 59)) return -3;  
        hour = h;  
        minute = m;  
        second = s;  
        return 0;  
    }  
  
    public String toString() {  
        return "Current time " + hour + ":" +  
            minute + ":" + second;  
    }  
}
```

Exemplu II

Cazuri excepționale

1. Nu mai am loc în șir
2. Numărul x nu e în șir
3. După x nu mai există nici un număr

```
class SirNumereReale {  
    private double[] sir;  
    private int nr = 0;  
    public SirNumereReale(int maxim) {  
        sir = new double[maxim];  
    }  
    public void adauga(double a) {  
        //Adauga numarul in sir  
    }  
    public double extrageDupa(double x) {  
        //Scoate si intoarce primul numar din sir aflat  
        //dupa prima pozitie la care se gaseste x  
    }  
}
```


Exemplu II

Dar ce întorc la extrageDupa
în cazurile 2,3 ?
(că NU avem la dispoziție o valoare
specială double)

```
class SirNumereReale {  
    private double[] sir;  
    private int nr = 0;  
    public SirNumereReale(int maxim) {  
        sir = new double[maxim];  
    }  
    public boolean adauga(double a) {  
        //Adauga numarul in sir  
        //Intoarce false daca nu mai e loc, altfel true  
    }  
    public double extrageDupa(double x) {  
        //Scoate si intoarce primul numar din sir aflat  
        //dupa prima pozitie la care se gaseste x  
    }  
}
```

Exemplu II

Ne trebuie un flag ...

```
class SirNumereReale {  
    private double[] sir;  
    private int nr = 0;  
    public SirNumereReale(int maxim) {  
        sir = new double[maxim];  
    }  
    public boolean adauga(double a) {  
        //Adauga numarul in sir  
        //Intoarce false daca nu mai e loc, altfel true  
    }  
    private boolean exceptie = false;  
    public boolean testExceptie() {  
        return exceptie;  
    }  
    public double extrageDupa(double x) {  
        //Scoate si intoarce primul numar din sir aflat  
        //dupa prima pozitie la care se gaseste x.  
        //flagul va fi 0 daca nu sunt probleme, -1 daca  
        //numarul nu e in sir, -2 daca nu exista nimic  
        //dupa el  
    }  
}
```

Exemplu II

Cerință : Un client trebuie să extragă și să calculeze media primelor 10 numere de după un număr x dat; dacă nu sunt 10 numere după x, se elimină numerele și se întoarce zero

```
class Utilitar {  
    public static double medie(double x,  
        SirNumereReale sir) {  
        double medie = 0;  
        for(int i = 0; i < 10; i++) {  
            double tmp = sir.extrageDupa(x);  
            if(!sir.testExceptie()) {  
                medie += tmp;  
            } else {  
                medie = 0;  
                break;  
            }  
        }  
        return medie / 10;  
    }  
}
```

Dar dacă x nu e în șir ?

```
class SirNumereReale {  
    private double[] sir;  
    private int nr = 0;  
    public SirNumereReale(int maxim) {  
        sir = new double[maxim];  
    }  
    public boolean adauga(double val) {  
        //Adauga numarul val la sfârșitul șirului  
        //Intoarce true dacă s-a adăugat, false altfel  
    }  
}
```

Apelantul **NU** poate fi obligat să facă verificări (ce să mai vorbim de a-l atenționa automat de situații speciale distincte)

Codul pentru cazul uzual se **mixează** cu codul tratării situațiilor excepționale

B

Mecanismul excepțiilor

Dr. Petru Florin Mihailescu

Definirea excepțiilor

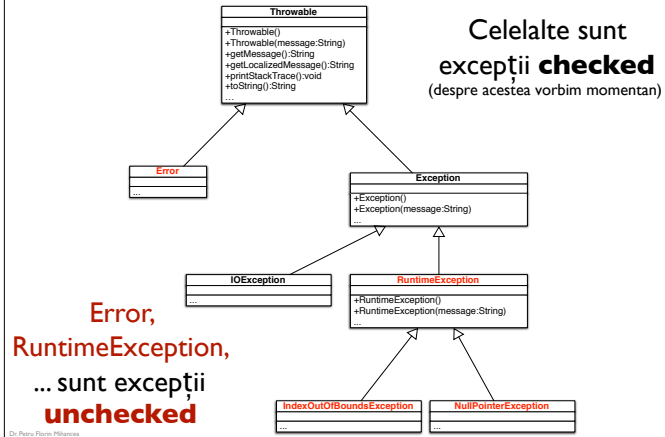
Sunt obiecte :)

clasa lor extinde o clasă specială din biblioteca Java
(uzual **clasa Exception**)
în rest sunt clase normale (ex. obiectele se crează la fel)

Pentru ceasul la care dorim să-i
setăm timpul, definim excepțiile
InvalidHour, InvalidMinute,
InvalidSecond

```
class InvalidHour extends Exception {  
    //Se practica sa ai ambii constructori  
    public InvalidHour() {}  
    public InvalidHour(String details) {super(details);}  
}  
class InvalidMinute extends Exception {  
    public InvalidMinute(String details) {super(details);}  
}  
class InvalidSecond extends Exception {  
    public InvalidSecond() {}  
}
```

Ierarhia (simplificată) de excepții



Emiterea excepției

```
class Clock {  
    private int hour, minute, second;  
    public Clock() {  
        hour = minute = second = 0;  
    }  
    public void setTime(int h, int m, int s) {  
        if((h < 0) || (h > 23)) throw new InvalidHour();  
        if((m < 0) || (m > 59)) throw new InvalidMinute("Minute " + m + " invalid!");  
        if((s < 0) || (s > 59)) throw new InvalidSecond();  
        hour = h;  
        minute = m;  
        second = s;  
    }  
    public String toString() {  
        return "Current time " + hour + ":" + minute + ":" + second;  
    }  
}
```

throw

simplistic, poate fi văzut ca un fel de return ... dar nu exagerați cu această comparație
(ex. unde se revine în codul programului e altfel)

Dr. Petru Florin Mihailescu

La compilare (verificări statice) ...

```
class Clock {  
    private int hour, minute, second;  
    public Clock() {  
        hour = minute = second = 0;  
    }  
    public void setTime(int h, int m, int s) {  
        if((h < 0) || (h > 23)) throw new InvalidHour();  
        if((m < 0) || (m > 59)) throw new InvalidMinute("Minute " + m + " invalid!");  
        if((s < 0) || (s > 59)) throw new InvalidSecond();  
        hour = h;  
        minute = m;  
        second = s;  
    }  
    public String toString() {  
        return "Current time " + hour + ":" + minute + ":" + second;  
    }  
}
```

Compilatorul devine extrem de vigilent la checked exceptions

Clock.java:12: error: unreported exception InvalidHour; must be caught or declared to be thrown
 if((h < 0) || (h > 23)) throw new InvalidHour();
 ^

Clock.java:13: error: unreported exception InvalidMinute; must be caught or declared to be thrown
 if((m < 0) || (m > 59)) throw new InvalidMinute("Minute " + m + " invalid!");
 ^

Clauza throws

```
class Clock {
    private int hour, minute, second;
    public Clock() {
        hour = minute = second = 0;
    }
    public void setTime(int h, int m, int s) throws InvalidHour, InvalidMinute,
        InvalidSecond {

        if((h < 0) || (h > 23)) throw new InvalidHour();
        if((m < 0) || (m > 59)) throw new InvalidMinute("Minute " + m + " invalid!");
        if((s < 0) || (s > 59)) throw new InvalidSecond();
        hour = h;
        minute = m;
        second = s;
    }
    public String toString() {
        return "Current time " + hour + ":" + minute + ":" + second;
    }
}
```

throw

- specifică ce fel de tipuri de excepții poate emite metoda (seamnă cu tipul returnat de o metodă)
- metoda va putea emite fără erori de compilare orice excepție de un tip/subtip de-a celor puse în clază

La compilare (verificări statice) ...

```
class Ceasornicar {  
    public void regleaza(Clock c, int h, int m, int s) {  
        c.setTime(h, m, s);  
    }  
  
    public static Clock creezaCeasReglatDeLaTastatura() {  
        Ceasornicar om = new Ceasornicar();  
        Clock c = new Clock();  
        int h = 0, m = 0, s = 0;  
        //Citeste de la tastatura h, m, s  
        om.regleaza(c, h, m, s);  
        return c;  
    }  
  
    public static void main(String argv[]) {  
        System.out.println(Ceasornicar.creezaCeasReglatDeLaTastatura());  
    }  
}
```

Programatorul trebuie să
decidă ce face (tratează
excepția
ori transmite excepția
apelantului)

Ceasornicar.java:6: error: unreported exception InvalidHour; must be
caught or declared to be thrown
c.setTime(h, m, s);
^
1 error

La compilare (verificări statice) ...

```
class Ceasornicar {  
  
    public void regleaza(Clock c, int h, int m, int s) throws InvalidHour {  
        c.setTime(h, m, s);  
    }  
  
    public static Clock creazaCeasReglatDeLaTastatura() {  
        Ceasornicar om = new Ceasornicar();  
        Clock c = new Clock();  
        int h = 0, m = 0, s = 0;  
        //Citește de la tastatura h, m, s  
        om.regleaza(c, h, m, s);  
        return c;  
    }  
  
    public static void main(String argv[]) {  
        System.out.println(Ceasornicar.creazaCeasReglatDeLaTastatura());  
    }  
}
```

Putem lăsa să se propage mai departe în apelantul metodei în care suntem (adică din reglează să meargă în apelantul ei); acțiune uzuală dacă în contextul curent nu știm cum să tratăm excepția

La compilare (verificări statice) ...

```
class Ceasornicar {  
  
    public void regleaza(Clock c, int h, int m, int s) throws InvalidHour {  
        c.setTime(h, m, s);  
    }  
  
    public static Clock creazaCeasReglatDeLaTastatura() {  
        Ceasornicar om = new Ceasornicar();  
        Clock c = new Clock();  
        int h = 0, m = 0, s = 0;  
        //Citește de la tastatura h, m, s  
        om.regleaza(c, h, m, s);  
        return c;  
    }  
  
    public static void main(String argv[]) {  
        System.out.println(Ceasornicar.creazaCeasReglatDeLaTastatura());  
    }  
}
```

Putem lăsa să se propage mai departe excepția dacă
suntem (adică din reglează să mediatizăm excepția) sau să
uzuală dacă în contextul curent nu se poate face altceva
deci să tratăm excepția

Trebuie decis pentru toate excepțiile
Ceasornicar.java:6: error: unreported exception InvalidMinute; must
be caught or declared to be thrown
c.setTime(h, m, s);
^
error

La compilare (verificări statice) ...

```
class Ceasornicar {  
  
    public void regleaza(Clock c, int h, int m, int s) throws InvalidHour, InvalidMinute, InvalidSecond {  
        c.setTime(h, m, s);  
    }  
  
    public static Clock creazaCeasReglatDeLaTastatura() {  
        Ceasornicar om = new Ceasornicar();  
        Clock c = new Clock();  
        int h = 0, m = 0, s = 0;  
        //Citeste de la tastatura h, m, s  
        om.regleaza(c, h, m, s);  
        return c;  
    }  
  
    public static void main(String argv[]) {  
        System.out.println(Ceasornicar.creazaCeasReglatDeLaTastatura());  
    }  
}
```

Ceasornicar.java:14: error: unreported exception InvalidHour; must be caught or declared to be thrown
 om.regleaza(c,h,m,s);
 ^
1 error

Interceptarea excepțiilor

```
...
try {
... cod ce tratează cazul uzual dar pe parcursul căreia ar putea apare excepții
} catch(Tip1 e) {
//Cod dedicat tratării tipului de excepție Tip1
//sau de orice subtip de-al său. Argumentul e referă excepția interceptată
} catch(Tip2 | Tip3 e) {
//Cod dedicat tratării tipului de excepție Tip2 și Tip3
//sau de orice subtipuri de-a lor (posibil din Java 1.7)
}
...
} catch(TipN e) {
//Pot fi oricâte catch-uri chiar și nici unul
} finally {
//Opțională. Conține cod ce se execută pe orice cale (în orice situație) am ieși din bloc
}
...
```

Interceptarea excepțiilor

```
...  
try {  
    ... cod ce tratează cazul uzual dar pe parcursul căreia ar putea apare excepții  
} catch(Tip1 e) {  
    //Cod dedicat tratării tipului de excepție Tip1  
    //sau de orice subtip de-al său. Argumentul e referă excepția interceptată  
} catch(Tip2 | Tip3 e) {  
    //Cod dedicat tratării tipului de excepție Tip2 și Tip3  
    //sau de orice subtipuri de-a lor (posibil din Java 1.7)  
}  
...  
} catch(TipN e) {  
    //Pot fi oricâte catch-uri chiar și nici unul  
} finally {  
    //Opțională. Conține cod ce se execută  
}  
...
```

Observație
Este eroare de compilare dacă:
I) pe o ramură catch se prinde o excepție care a fost deja
prinsă (inclusiv printr-un supertip) de un catch anterior
II) pe o ramură catch se prinde o excepție subtip de-a lui
Exception și care nu poate apare în secțiunea try
(evident nepropagată mai departe)

La compilare (verificări statice) ...

```
class Ceasornicar {
    public void regleaza(Clock c, int h, int m, int s) throws InvalidHour, InvalidMinute, InvalidSecond {
        c.setTime(h, m, s);
    }
    public static Clock creazaCeasReglatDeLaTastatura() {
        Ceasornicar om = new Ceasornicar();
        Clock c = new Clock();
        int h = 0, m = 0, s = 0;
        boolean problem;
        do {
            problem = false;
            try {
                BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
                h = Integer.parseInt(bf.readLine());
                //...
                om.regleaza(c, h, m, s);
            } catch (InvalidHour e) {
                problem = true;
            } catch (InvalidMinute | InvalidSecond e) {
                problem = true;
            }
        } while(problem);
        return c;
    }
    public static void main(String argv[]) {
        System.out.println(Ceasornicar.creazaCeasReglatDeLaTastatura());
    }
}
```

Ceasornicar.java:21: error: unreported exception IOException; must be caught or declared to be thrown
String hourString = bf.readLine();

La compilare (verificări statice) ...

```
class Ceasornicar {
    public void regleaza(Clock c, int h, int m, int s) throws InvalidHour, InvalidMinute, InvalidSecond {
        c.setTime(h, m, s);
    }
    public static Clock creazaCeasReglatDeLaTastatura() throws IOException {
        Ceasornicar om = new Ceasornicar();
        Clock c = new Clock();
        int h = 0, m = 0, s = 0;
        boolean problem;
        do {
            problem = false;
            try {
                BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
                h = Integer.parseInt(bf.readLine());
                //...
                om.regleaza(c, h, m, s);
            } catch (InvalidHour e) {
                problem = true;
            } catch (InvalidMinute | InvalidSecond e) {
                problem = true;
            }
        } while(problem);
        return c;
    }
    public static void main(String argv[]) throws IOException {
        System.out.println(Ceasornicar.creazaCeasReglatDeLaTastatura());
    }
}
```

Quiz

Ceva problemă cu codul de mai jos ?

```
Clock c = new Clock();
try {
    c.setTime(23, 0, 0);
} catch (Exception e) {
    System.err.println(e);
} catch (InvalidHour e) {
    System.err.println(e);
}
```

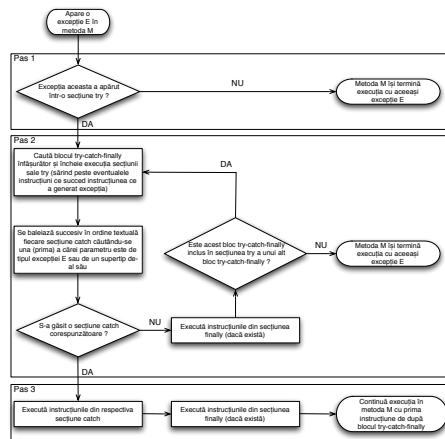
Nu compilează pentru că
InvalidHour e subtip Exception și
Exception e prinsă de un catch
anterior. Invers ar fi compilabil.

... dar de ce compilează și nu am prins
InvalidMinute de exemplu ?

```
Clock c = new Clock();
try {
    c.setTime(23, 0, 0);
} catch (InvalidHour e) {
    System.err.println(e);
} catch (Exception e) {
    System.err.println(e);
}
```

Sunt prinse de ultimul catch
(Exception fiind supertip pentru ele).
Dacă ar lipsii ultimul catch da, ar fi
eroare de compilare.

La execuție(dinamic) (aprox., pt. o metodă)



```
class Clock {
    private int hour, minute, second;
    public Clock() {
        hour = minute = second = 0;
    }
    public void setTime(int h, int m, int s) throws InvalidHour, InvalidMinute,
        InvalidSecond {
        if((h < 0) || (h > 23)) throw new InvalidHour();
        if((m < 0) || (m > 59)) throw
            new InvalidMinute("Minute " + m + " invalid!");
        if((s < 0) || (s > 59)) throw new InvalidSecond();
        hour = h;
        minute = m;
        second = s;
    }
    public String toString() {
        return "Current time " + hour + ":" + minute + ":" + second;
    }
}
```

```
    } catch (InvalidHour e) {
        problem = true;
    } catch (InvalidMinute | InvalidSecond e) {
        problem = true;
    }
```

Diagram:

Start: Apare o excepție E în metoda M

Decision: Pas 1: Excepția această a apărut într-o secțiune try?

Yes (DA): Metoda M își termină execuția cu excepția E

No (NU): Metoda M își termină execuția cu excepția E

Exemplu

setTime-{this=0x552,h=25,m=0,s=0}
regleaza - {c=0x552,h=25,m=0,s=0}
creazaCearReglatDeLaTastatura - {...}
main - {argv = []}

Stiva de execuție

Presupunem că s-a citit ora 25, minutul 0 și secunda 0

```

graph TD
    Start([Apăre o excepție E în metoda M]) --> Pas1[Pas 1]
    Pas1 --> Dec1{Excepția această e apărut  
încă o dată în metoda M?}
    Dec1 -- DA --> End1([Metoda M s-a terminat  
cu excepția E])
    Dec1 -- NU --> End2([Metoda M s-a terminat  
execuția cu excepția E])
    End2 --> End3([Metoda M s-a terminat  
execuția cu excepția E])
    
```

regleaza - {c=0x552,h=25,m=0,s=0}
creazaCearReglatDeLaTastatura - {...}
main - {argv = []}

Stiva de execuție

```
main - {argv = []}
```

Presupunem că s-a citit ora 25,
minutul 0 și secunda 0

```

graph TD
    Start([Apere o exceptie E in metoda M]) --> Pas1[Pas 1]
    Pas1 --> Decision{Exceptia aceasta a apărut într-o secțiune try?}
    Decision -- DA --> End([Metoda M își termină execuția cu aceeași excepție E])
    Decision -- Nu --> X((X))
  
```

creazaCeasReglatDeLaTastatura - {...}
main - {argv = []}

Stiva de execuție

Presupunem că s-a citit ora 25,
minutul 0 și secunda 0

```

class Ceasornicar {
    public void regleaza(Clock c, int h, int m, int s)
        throws InvalidHour, InvalidMinute, InvalidSecond {
        c.setTime(h, m, s);
    }
    public static Clock creezaCeasReglatDeLaTastatura()
        throws IOException {
        Ceasornicar om = new Ceasornicar();
        Clock c = new Clock();
        int h = 0, m = 0, s = 0;
        boolean problem;
        do {
            problem = false;
            try {
                BufferedReader bf = new
                BufferedReader(new InputStreamReader(System.in));
                h = Integer.parseInt(bf.readLine());
            }
            //...
            om.regleaza(c, h, m, s);
            System.out.println("Done!");
        } catch (InvalidHour e) {
            problem = true;
        } catch (InvalidMinute | InvalidSecond e) {
            problem = true;
        }
        System.out.println("Problem?: " + problem);
        } while(problem);
        return c;
    }
    public static void main(String argv[]) throws IOException {
        System.out.println(Ceasornicar.creezaCeasReglatDeLaTastatura());
    }
}

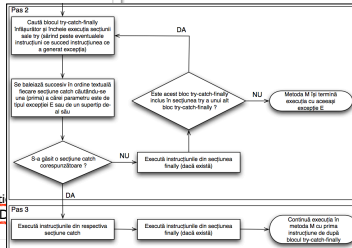
```

Exemplu

creazaCeasReglatDeLaTastatura - {...}

main - {argv = []}

Stiva de execuție



```

class Ceasornicar {
    public void regleaza(Clock c, int h, int m, int s)
        throws InvalidHour, InvalidMinute, InvalidSecond {
        c.setTime(h, m, s);
    }

    public static Clock creazaCeasReglatDeLaTastatura()
        throws IOException {
        Ceasornicar om = new Ceasornicar();
        Clock c = new Clock();
        int h = 0, m = 0, s = 0;
        boolean problem;
        do {
            problem = false;
            try {
                BufferedReader bf = new
datorită
buciei!
                BufferedReader(new InputStreamReader(System.in));
                h = Integer.parseInt(bf.readLine());
                //...
                om.regleaza(c, h, m, s);
                System.out.println("Done!");
            } catch (InvalidHour e) {
                problem = true;
            } catch (InvalidMinute | InvalidSecond e) {
                problem = true;
            }
            System.out.println("Problem?: " + problem);
        } while (problem);
        return c;
    }

    public static void main(String argv[]) throws IOException {
        System.out.println(Ceasornicar.creazaCeasReglatDeLaTastatura());
    }
}

```

Exemplu

```
creazaCeasReglatDeLaTastatura - {...}
```

```
main - {argv = []}
```

Stiva de execuție

Se continuă execuția normală începând cu prima instrucțiune de după try-catch-finally

Quiz

Presupunem că am fi scris programul să propage InvalidHour până înafara lui main. Ce s-ar fi întâmplat dacă dădeam o oră greșită?

Programul se oprește cu mesajul:

```
Exception in thread "main" java.lang.IllegalThreadStateException: (Click Java 12)  
at cseascepticii.Ceasnicii.cor.mergemacasa(Ceasnicii.cor.java:10)  
at cseascepticii.Ceasnicii.cor.creareceasniciladeafostator(Ceasnicii.cor.java:35)  
at cseascepticii.Ceasnicii.cor.malin(Ceasnicii.cor.java:35)
```

Excepții unchecked

```

class Ceasornicar {
    public void regleaza(Clock c, int h, int m, int s)
        throws InvalidHour, InvalidMinute, InvalidSecond {
        c.setTime(h, m, s);
    }
    public static Clock creazaCeasReglatDeLaTastatura()
        throws IOException {
        Ceasornicar om = new Ceasornicar();
        Clock c = new Clock();
        int h = 0, m = 0, s = 0;
        boolean problem;
        do {
            problem = false;
            try {
                BufferedReader bf = new
                    BufferedReader(new InputStreamReader(Sy
                        h = Integer.parseInt(bf.readLine());
                //...Integer.parseInt
                om.regleaza(c, h, m, s);
                System.out.println("Done!");
            } catch (InvalidHour e) {
                problem = true;
            } catch (InvalidMinute | InvalidSecond e) {
                problem = true;
            }
            System.out.println("Problem?: " + problem);
        } while (problem);
        return c;
    }
    public static void main(String argv[]) throws IOException {
        System.out.println(Ceasornicar.creazaCeasReglatDeLaTastatura());
    }
}

```

Ce se întâmplă dacă de la tastatură dăm o oră care **NU** e număr ?

Programul se oprește cu mesajul:

```

Exception in thread "main" java.lang.NumberFormatException: For input string: "90"
at java.lang.Integer.parseInt(Integer.java:592)
at java.lang.Integer.parseInt(Integer.java:527)
at Ceasornicar.creazaCeasReglatDeLaTastatura(Ceasornicar.java:35)
at Ceasornicar.main(Ceasornicar.java:55)

```

Dacă Integer.parseInt se poate termina cu excepția **NumberFormatException** de ce a compilat programul chiar dacă **NU** am tratat ?

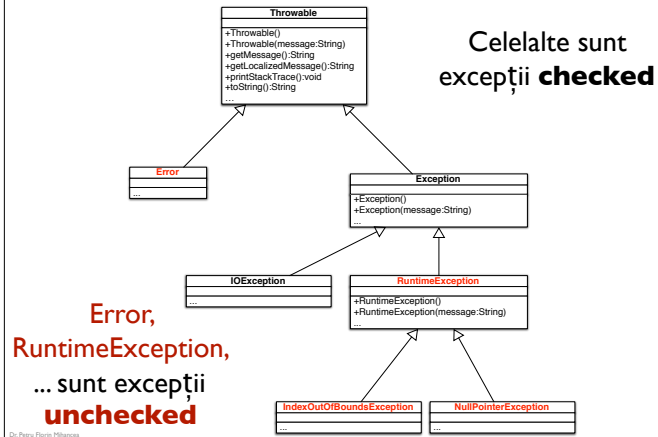
NumberFormatException este o excepție neverificată (unchecked).

```
Integer c = null;  
System.out.println(c.intValue());
```

Programul se oprește cu mesajul:
Exception in thread "main"
at org.qlizz.Quizz2.main(Quizz2.java:4)

NullPointerException este o
excepție neverificată (unchecked).

Ierarhia (simplificată) de excepții



Dr. Petru Florin Păvăleanu

Excepțiile neverificate (unchecked)

Subclase de-a lui **Error** sau **RuntimeException**

Se emit tot cu **throw**

Se prind tot cu **try-catch-finally**

Se comportă/propagă la execuție la fel

**... dar compilatorul NU
face verificări pentru ele**

ex. dacă s-a tratat sau nu într-o metodă,
dacă s-a pus clauza **throws**, etc.

D

Particularități în contextul excepțiilor

Dr. Petru Florin Mihailescu

La overriding

Când o metodă X suprascrie/overrides o metodă Y dintr-o clasă de bază, throws-ul metodei X poate conține doar excepții checked specificate în clauza throws de la Y (sau subtipuri de-a lor)

```
class ExceptionS extends Exception{};
class ExceptionA extends ExceptionS{};
class ExceptionB extends ExceptionS{};

abstract class BaseClass {
    public abstract void method1();
    public void method2() {
    }
    public abstract void method3() throws ExceptionS;
    public void method4() throws ExceptionA {
    }
    public void method5() throws ExceptionB {
    }
}
```

```
class SubClass extends BaseClass {
    //Eroare compilare
    public void method1() throws ExceptionS {
        throw new ExceptionS();
    }
    //Eroare compilare
    public void method2() throws ExceptionA {
        throw new ExceptionA();
    }
    public void method3() throws ExceptionA {
        throw new ExceptionA();
    }
    //Eroare compilare
    public void method4() throws ExceptionB {
        throw new ExceptionB();
    }
    public void method5() {
    }
}
```


La overriding

Când o metodă X suprascrie/overrides o metodă Y dintr-o clasă de bază, throws-ul metodei X poate conține doar excepții checked specificate în clauza throws de la Y (sau subtipuri de-a lor)

```
class ExceptionS extends Exception{};
class ExceptionA extends ExceptionS{};
class ExceptionB extends ExceptionS{};

interface AnInterface {
    public void method6() throws ExceptionA;
}
abstract class ABaseClass {
    public abstract void method6() throws ExceptionB;
}
```

```
class ASubClass1
    extends ABaseClass implements AnInterface {
    //Eroare de compilare
    public void method6() throws ExceptionA {}
}
```

```
class ASubClass2
    extends ABaseClass implements AnInterface {
    //Eroare de compilare
    public void method6() throws ExceptionB {}
}
```

```
class ASubClass3
    extends ABaseClass implements AnInterface {
    //Eroare de compilare
    public void method6() throws ExceptionA,ExceptionB {}
}
```

```
class ExceptionS extends Exception{};
class ExceptionA extends ExceptionS{};
class ExceptionB extends ExceptionS{};
```

```
abstract class BaseClass {
    public abstract void method1();
    public void method2() {
    }
    public abstract void method3() throws ExceptionS;
    public void method4() throws ExceptionA {
    }
    public void method5() throws ExceptionB {
    }
}
```

```
class AClient {
    void MyMethod(BaseClass a) {
        a.method1();
        try {
            a.method3();
        } catch (ExceptionS e) {
            ...
        }
    }
}
```

De ce ?

```
class SubClass extends BaseClass {
    //Eroare compilare
    public void method1() throws ExceptionS {
        throw new ExceptionS();
    }
    //Eroare compilare
    public void method2() throws ExceptionA {
        throw new ExceptionA();
    }
    public void method3() throws ExceptionA {
        throw new ExceptionA();
    }
    //Eroare compilare
    public void method4() throws ExceptionB {
        throw new ExceptionB();
    }
    public void method5() {
    }
}
```

Dacă s-ar permite altfel, un client al clasei de bază ar putea primi la rulare excepții nenumționate de metodele clasei de bază

La constructori

```
class ExceptionS extends Exception{};
class ExceptionA extends ExceptionS{};
class ExceptionB extends ExceptionS{};

class BaseClass {
    public BaseClass() throws ExceptionA {
    }
}
```

```
class SubClass extends BaseClass {
```

```
class SubClass extends BaseClass {
    public SubClass() {
        try {
            super();
        } catch (ExceptionA x) {
        }
    }
}
```



```

} catch (ExceptionA x) {
}
}
}

```

MyConstructorRestrictions.java:15: error: call to super must be first statement in constructor

```

    super();
    ^

```

1 error

```
MyConstructorRestrictions.java:12: error: unreported
exception ExceptionA in default constructor
class SubClass extends BaseClass {}
^
1 error
```

```
class SubClass extends BaseClass {
    public SubClass() throws ExceptionA, ExceptionB {
    }
}
```

Doar cu throws pt. ExceptionA. În plus
putem adăuga și alte excepții verificate

```
class ExceptionS extends Exception{};
class ExceptionA extends ExceptionS{};
class ExceptionB extends ExceptionS{};
class Quiz3 {
    public void aMethod() throws ExceptionS {
        // throw new ExceptionA();;
    }
    public void aCaller() {
        try {
            this.aMethod(); // Eroare de compilare. De ce?
        } catch (ExceptionA a) {
            ...
        } catch (ExceptionB a) {
            ...
        }
    }
}
```

Pt. că aMethod poate emite orice fel de **Exceptions** (inclusiv subtipuri), deci ca să putem compila trebuie să avem ca ultim catch și Exceptions

E

Atenție

Dr. Petru Florin Mihailescu

finally

Se execută oricum am ieșii din try
normal, trecând printr-un catch și chiar și când excepția curentă nu e
prinsă de niciun catch

```
class MaiMareSauEgal extends Exception {  
    public MaiMareSauEgal() {super("Numar >= 0.5");}  
}  
class LaFinally {  
    public static void main(String[] args) {  
        try {  
            if(Math.random() >= 0.5) {  
                throw new MaiMareSauEgal();  
            }  
            System.out.println("Numar < 0.5");  
        } catch (MaiMareSauEgal e) {  
            System.out.println(e.getMessage());  
        } finally {  
            System.out.println("Gata!"); // Tot timpul apare pe ecran  
        }  
    }  
}
```

Reemiterea. Nu pierdeți **cauza**

```
class MyException extends Exception {  
    public MyException(Throwable a) {  
        super(a);  
    }  
    public MyException() {}  
}  
  
public class Lost {  
    public static void test() throws MyException {  
        throw new MyException();  
    }  
    public static void main(String[] args) throws MyException {  
        try {  
            test();  
        } catch (MyException a) {  
            throw new MyException();  
        }  
    }  
}
```

La ieșire se pierde **cauza origină**
Exception în throw "new" cauză: MyException
at atentie.Lost.main(Lost.java:10)

Trebuie **throws** (sau alt try-catch-finally
înfășurător) pt. că noua excepție nu se
tratează în try-catch-finally existent

Reemiterea. Nu pierdeți cauza

```
class MyException extends Exception {  
    public MyException(Throwable a) {  
        super(a);  
    }  
    public MyException() {}  
}  
  
public class Lost {  
    public static void test() throws MyException {  
        throw new MyException();  
    }  
    public static void main(String[] args) throws MyException {  
        try {  
            test();  
        } catch (MyException a) {  
            throw new MyException(a);  
        }  
    }  
}
```

În al doilea caz avem:
Exception (in thread "auv") atentia.MyException
at atentia.Lost.test(Lost.java:13)
Caused by: atentia.Lost.test(Lost.java:13)
at atentia.Lost.main(Lost.java:17)

Utilizează:
throw a;
sau
throw new MyException(a);
//utilizăm un constructor
special din Exception/
Throwable

Se poate **pierde** o excepție

```
public static void test() throws MyException {  
    try {  
        throw new MyException();  
    } finally {  
        return;  
    }  
}
```

Atenție

În Java mecanismul excepțiilor e cu terminare

"Operația" ce a produs excepția **NU** se reia
singură;
noi ca programatori trebuie să scriem codul
corespunzător dacă vrem reluarea operației
(ex. probabil ceva buclă)

Nu ignora o excepție

```
try {  
    ...  
} catch(CevaExcepție e) { }
```