Përpunimi i tekstit dhe i fajleve

Faton Berisha

Fakulteti i Shkencave Kompjuterike
Universiteti i Prizrenit

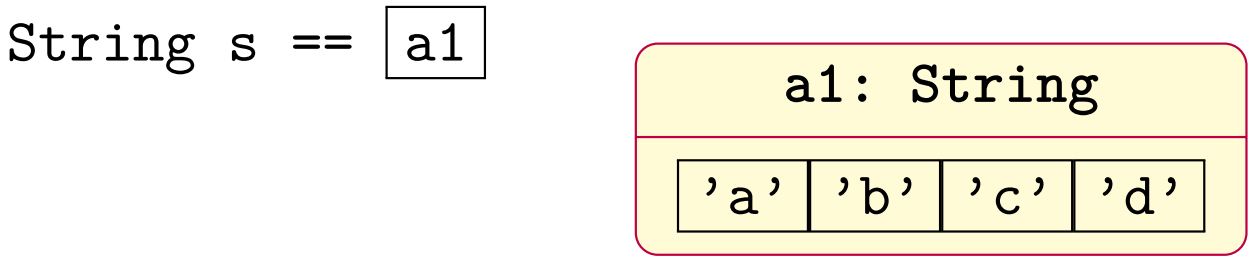
Qëllimet dhe objektivat

- Zbërthimi i një rreshti teksti që përmban të dhëna të ndryshme në pjesë përbërëse, të quajtura *tokenë*, me anë të tokenizuesve të stringjeve
- Leximi dhe shkrimi i rreshtave të tekstit në dhe nga fajla sekuencialë në memorie periferike
- Adaptimi i teknikave të njëjta për lexim rreshtash të tekstit nga dritarja komanduese
- Përdorimi i përpunuesve të gabimeve (exception handlers) për të përpunuar gabimet në përpunimin e fajlave

Përmbajtja

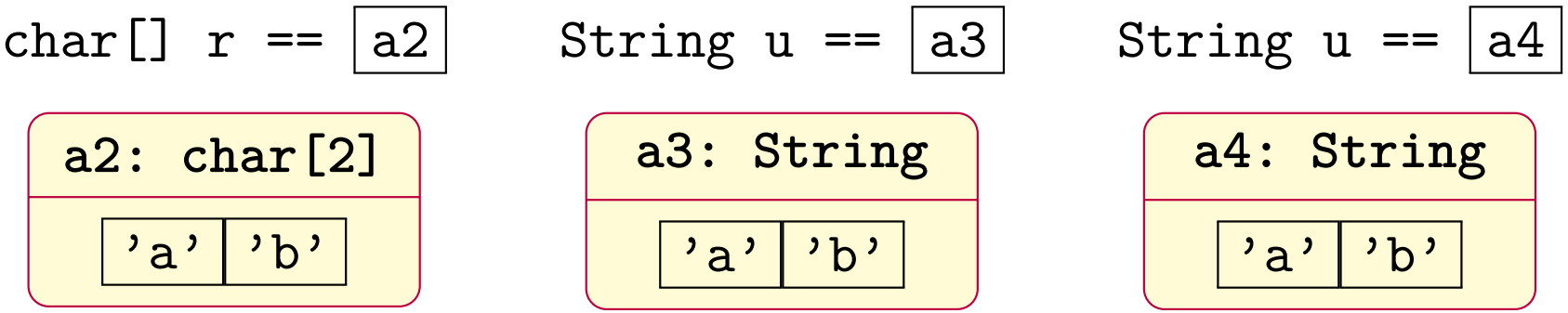
Stringjet janë objekte të pandryshueshme (immutable)

- Stringjet janë objekte, të cilat në Java paraqiten si vargje karakterësh të mbyllura brenda thonjëzave të dyfishta.
- Sintaksa e Java mundëson që me variablat e tipit `String` të punohet sikur me variabla të tipeve primitive.



Stringjet janë objekte

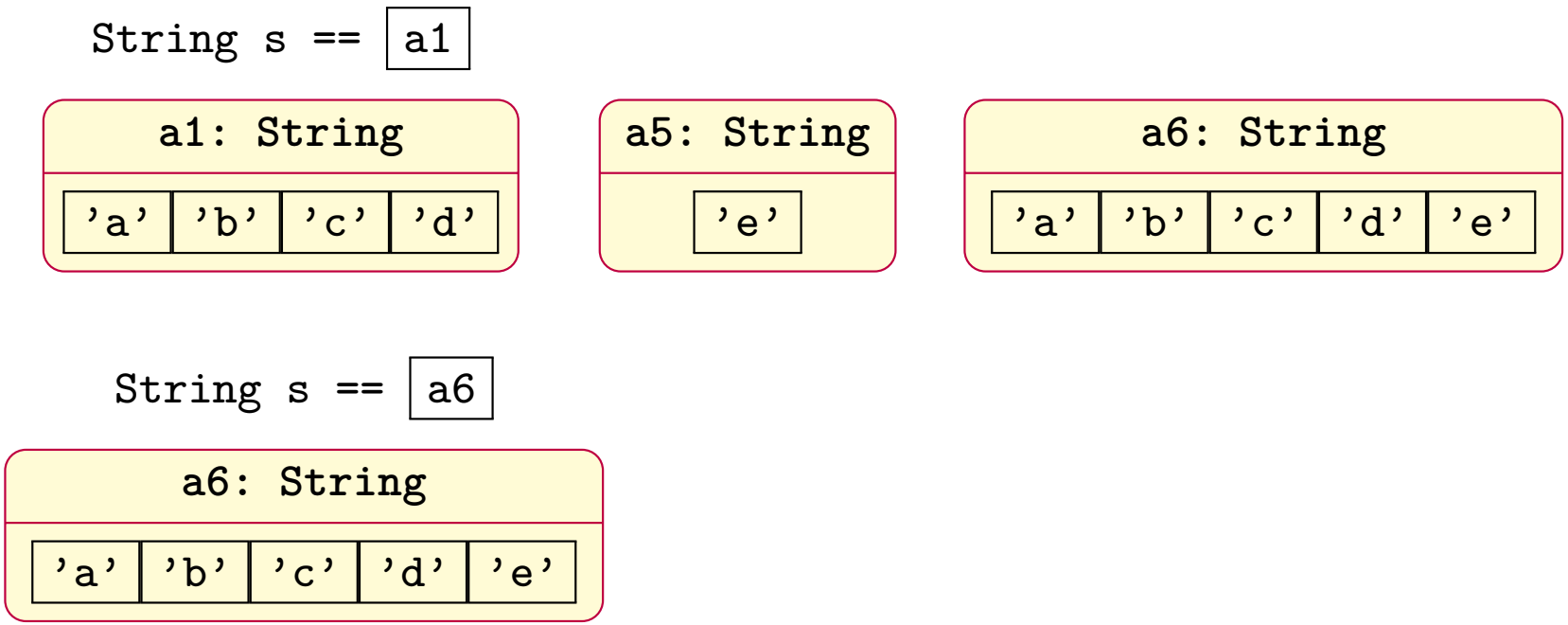
```
char[] r = new char[2];
r[0] = 'a';
r[1] = 'b';
String u = new String(r);
String v = "ab";
```



Stringjet janë të pandryshueshme (immutable)

- Java stringjet janë *të pandryshueshme* (*immutable*): kur një objekt `String` të jetë krijuar, përmbajtja e tij nuk mund të ndryshohet.

```
s = s + "e";
```



Klasa StringBuffer

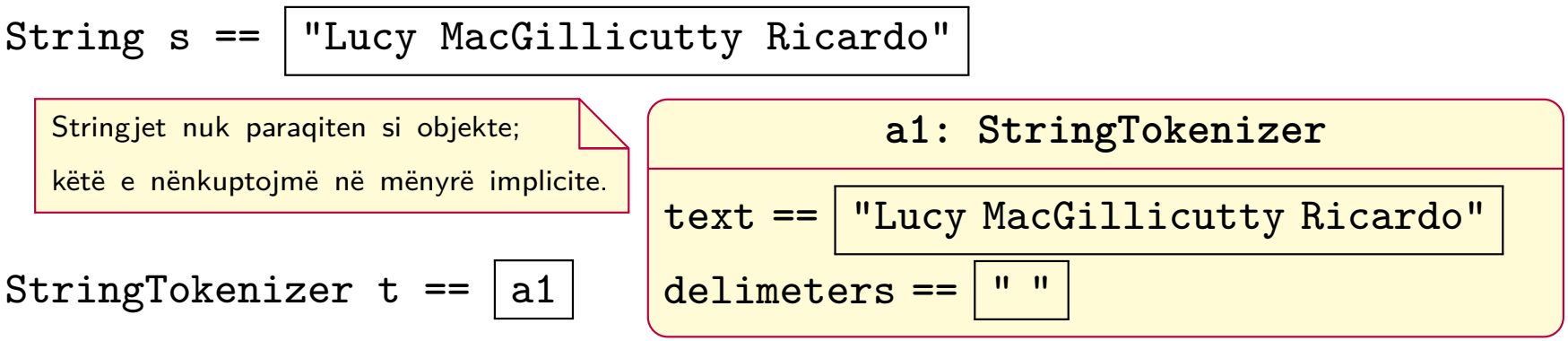
- Java ofron një version stringjesh të cilat janë të ndryshueshme (mutable). Klasa quhet `StringBuffer` dhe gjendet në pakon `java.lang`

```
StringBuffer sb = new StringBuffer(s);
sb.append("e");
```

Tokenizuesit e stringjeve

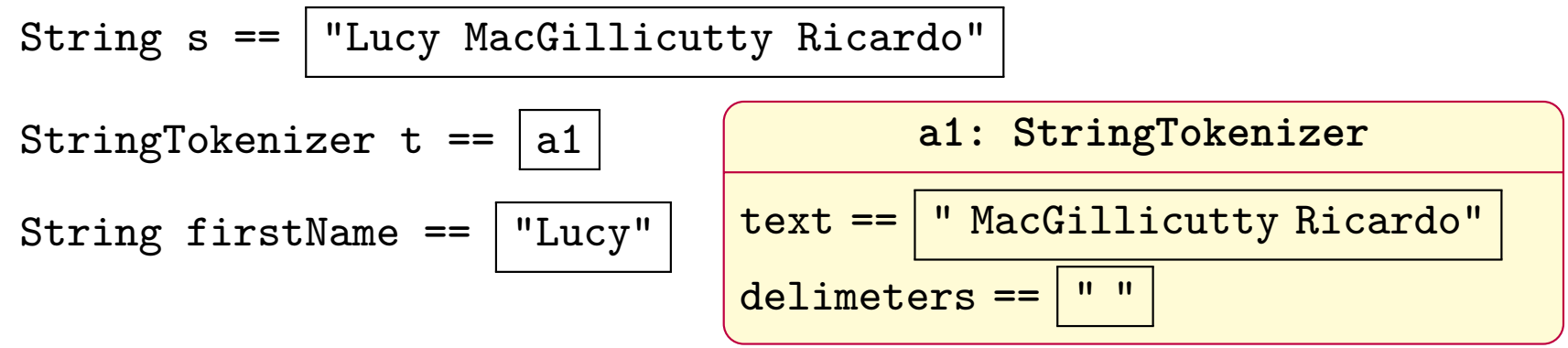
- *Token:* një „fjalë“ (njësi informacioni) brenda një stringu ose një rreshti teksti.
- *Delimetër* (*delimeter*): një karakter që përdoret për të ndarë tokenët në një string; p.sh., hapësira dhe shenjat e pikësimit.
- Klasa `StringTokenizer`, e cila gjendet në pakon `java.util`, përmban metoda për zbërthim të stringjeve në tokenë.

```
String s = "Lucy MacGillicutty Ricardo";
StringTokenizer t = new StringTokenizer(s, " ");
```



Tokenizuesit e stringjeve (Vazhdim)

```
String firstName = t.nextToken();
```




```
String middleName = t.nextToken();
String lastName = t.nextToken();
```

Klasa StringTokenizer

`class StringTokenizer`	
Konstruktori	
`StringTokenizer(String text, String delim)`	Konstrukton një tokenizues që ekstrakton tokenë nga `text` me të gjithë karakterët në `delim` si separatorë
`String nextToken()`	Largon të gjithë delimetrat. Kthen dhe largon vargun maximal jobosh të karakterëve deri në një delimetër ose në fund.
`String nextToken(String delim)`	Sikur `nextToken()`, por më parë reseton delimetrat e specifikuar.
`boolean hasMoreTokens()`	Kthen se a ka mbetur ndonjë token në stringun në tokenizuesin.
`int countTokens()`	Kthen numrin e tokenëve të mbetur në stringun në tokenizuesin.

Tabela. Metodat kryesore të klasës `StringTokenizer`

```
String s = "13.46 E";
StringTokenizer t = new StringTokenizer(s, ". E");
String euros = t.nextToken();
String cents = t.nextToken();
```

Fajlat sekuencialë

- *Fajl karakterësh* (*character file*): varg karakterësh të ruajtura në fajlin si varg bajtësh sipas një sistemi *enkodimi* (*encoding*).
- *Fajl binar* (*binary file*): varg bitësh.

- Një fajl *sekuencial* duhet të lexohet (ose shkruhet) nga fillimi deri në fund, në një drejtim.
- Një fajl *me qasje të çfarëdoshme* (*random access*) lejon lexim (ose shkrim) në ëfarëdo pozita brenda tij.

- Hapja e një fajli sekuencial për lexim (*input*): fajli lexohet në mënyrë sekuenciale dhe nuk mund të shkruhet.
- Hapja e një fajli sekuencial për shkrim (*output*): fajli shkruhet në mënyrë sekuenciale dhe nuk mund të lexohet.

Output në fajla sekuencialë

1. Konstruktohet një `FileWriter` objekt, i cili mban adresën fizike të fajlit të emërtuar (p.sh. `test.txt`) në memorien periferike dhe shkruan karakterët individualë në fajlin:

```
FileWriter w = new FileWriter("test.txt");
```

2. Kompozohet `FileWriter` me një `PrintWriter`, i cili ka metodat `print(`E`)` dhe `println(`E`)` për ta dërguar stringun E karakter për karakter në objektin `w`:

```
FilePrinter outFile = new FilePrinter(w);
```

Shembull aplikacioni

```
import java.io.*;

/** Shkruan tre rreshta tekst në fajlin test.txt */
public class Output1 {
  public static void main(String[] args) throws IOException {
    PrintWriter outFile
      = new PrintWriter(new FileWriter("test.txt"));
    outFile.println("Tungjatjeta!");
    outFile.print("Si keni");
    outFile.println(" qenë?");
    outFile.println(47 + 2);
    outFile.close();
  }
}
```

Input nga fajla sekuencialë

1. Konstruktohet një `FileReader` objekt, i cili mban adresën fizike të fajlit të emërtuar në memorien periferike dhe lexon karakterët individualë nga fajli:

```
FileReader r = new FileReader("test.txt");
```

2. Kompozohet `FileReader` me një `BufferedReader`, i cili ka metodën `readLine()` për leximin e një rreshti të plotë teksti:

```
BufferedReader inFile = new BufferedReader(r);
```

Aplikacion që kopjon një fajl

```
import javax.swing.*;
import java.io.*;
/** Kopjon përmbajtjen e një input fajli, me emër f të specifikuar
 * nga shfrytëzuesi, në output fajlin f.out */
public class CopyFile {
  public static void main(String[] args) throws IOException {
    String f = JOptionPane.showInputDialog("Emri i input fajlit:");
    // Konstrukto input view që lexon nga input fajli:
    BufferedReader inFile = new BufferedReader(new FileReader(f));
    // Konstrukto output view që shkruan në output fajlin:
    PrintWriter outFile = new PrintWriter(new FileWriter(f + ".out"));
    while (inFile.ready()) { // A ka më rreshta për të lexuar?
      String s = inFile.readLine();
      outFile.println(s);
    }
    inFile.close();
    outFile.close();
  }
}
```

Input sekuencial nga dritarja komanduese

1. Konstruktohet një `InputStreamReader` objekt nga `System.in`:

```
InputStreamReader r
  = new InputStreamReader(System.in);
```

2. Kompozohet `InputStreamReader` me një `BufferedReader`:

```
BufferedReader keyboard = new BufferedReader(r);
```

Versioni i modifikuar i aplikacionit MakeChange

```
import java.io.*;

/** Llogarit sasinë e metelikeve.
 *  Input: sasia e eurove dhe sasia e centëve nga konsola
 *  Output: sasitë e metelikëve */
public class MakeChange {
  public static void main(String[] args) throws IOException {
    BufferedReader keyboard
      = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("Sasia e eurove: ");
    int euros = new Integer(keyboard.readLine()).intValue();
    System.out.print("Sasia e centëve: ");
    int cents = new Integer(keyboard.readLine()).intValue();
```

Versioni i modifikuar i MakeChange (Vazhdim)

```
    if ((euros < 0) || (cents < 0) || (cents > 99)) {
      System.out.println("Gabim në hyrje " + euros + " " + cents);
    } else {
      int money = euros * 100 + cents;
      System.out.println("Gjysma = " + (money / 50));
      money = money % 50;
      System.out.println("Njëzetqindarka = " + (money / 20));
      money = money % 20;
      System.out.println("Dhjetëqindarka = " + (money / 10));
      money = money % 10;
      System.out.println("Pesëqindarka = " + (money / 5));
      money = money % 5;
      System.out.println("Qindarka = " + money);
    }
  }
}
```

Studim rasti: Përpunimi i një liste pagash

`class PayrollReader`	Lexon regjistrime nga një fajl sekuecial.
`PayrollReader(String f)`	Konstrukton lexuesin që lexon nga fajli `f`
`boolean getNextRecord()`	Përpiqet të lexojë regjistrimin e ardhshëm nga fajli.
`String nameOf()`	Kthen emrin e punëtorit nga regjistrimi aktual.
`int hoursOf()`	Kthen kthen numrin e orëve të punës.
`double payrateOf()`	Kthen çmimin e orës së punës.
`close()`	Mbyll fajlin listës së pagave.

Tabela. API i klasës `PayrollReader`

- Input view komponenta `PayrollReader` e aplikacionit nxjerr emrin, numrin e orëve të punës dhe çmimin e orës nga secili regjistrim në një input fajl të formatizuar:

```
Fisnik Bardhi|40|28.25
Tringa Kosova|42|24.50
Filane Ziu|18|18.00
```

Aplikacioni Payroll

```
import javax.swing.*;
/** Printon një fajl pagash nga një input fajl me listë pagash. */
public class Payroll {
  public static void main(String[] args) {
    String inFile = JOptionPane.showInputDialog("Input lista e pagave:");
    String outFile = JOptionPane.showInputDialog("Output lista e çeqeve:");
    if (inFile != null && outFile != null) {
      processPayroll(inFile, outFile);
    }
    System.out.println("Fund");
  }
  private static void processPayroll(String in, String out) {
    PayrollReader reader = new PayrollReader(in);
    PayrollWriter writer = new PayrollWriter(out);
    while (reader.getNextRecord()) {
      double pay = reader.hoursOf() * reader.payrateOf();
      writer.printCheck(reader.nameOf(), pay);
    }
    reader.close();
    writer.close();
  }
}
```

Input view PayrollReader e aplikacionit

```
import java.io.*;
import java.util.*;
/** Lexon regjistrime nga një fajl sekuecial */
public class PayrollReader {
  private BufferedReader inFile; // Adresa e input fajlit
  // Emri, numri i orëve dhe çmimi për orë i regjistrimit aktual
  private String name;
  private int hours;
  private double payrate;

  public PayrollReader(String file) {
    try {
      inFile = new BufferedReader(new FileReader(file));
    } catch (Exception e) {
      System.out.println("Gabim PayrollReader: emër fajli jokorrekt: "
          + file + " Fund!");
      throw new RuntimeException(e.toString());
    }
  }
```

Input view PayrollReader (Vazhdim)

```
public String nameOf() {
  return name;
}

public int hoursOf() {
  return hours;
}

public double payrateOf() {
  return payrate;
}

/** Mbyll input fajlin */
public void close() {
  try {
    inFile.close();
  } catch (IOException e) {
    System.out.println(
        "Vërejtje PayrollReader: dështoi mbyllja e fajlit");
  }
}
```

Input view PayrollReader (Vazhdim)

```
/** Përpiqet të lexojë një regjistrim të ri
 * @return se a ësthë lexuar dhe është gadi të përpunohet regjistrimi */
public boolean getNextRecord() {
  boolean result = false;
  try {
    if (inFile.ready()) {
      String line = inFile.readLine();
      StringTokenizer t = new StringTokenizer(line, "|");
      String s = t.nextToken().trim();
      if (t.countTokens() == 2) { // Orët dhe çmimi?
        name = s;
        hours = new Integer(t.nextToken().trim()).intValue();
        payrate = new Double(t.nextToken().trim()).doubleValue();
        result = true;
      } else {
        throw new RuntimeException(line);
      }
    }
```

Input view PayrollReader (Vazhdim)

```
    }  catch (IOException e) {
      System.out.println("Gabim PayrollReader: " + e.getMessage());
    } catch (RuntimeException e) {
      System.out.println("Gabim PayrollReader: format jokorrekt: "
          + e.getMessage() + " Kapërcim regjistrimi");
      result = getNextRecord(); // Përpiqu sërish
    }
    return result;
  }
}
```

Output view PayrollWriter i aplikacionit

```
import java.io.*;

/* Shruan regjistrime çeqesh në një fajl sekuecial. */
public class PayrollWriter {
  PrintWriter outFile; // Adresa e output fajlit

  public PayrollWriter(String file) {
    try {
      outFile = new PrintWriter(new FileWriter(file));
    } catch (IOException e) {
      System.out.println("Gabim PayrollWriter: " + file + " Fund!");
      throw new RuntimeException(e.toString());
    }
  }
```

Output view PayrollWriter (Vazhdim)

```
  /** Shkruan regjistrimin e ri në output fajlin
   * @param name emri i punëtorit
   * @param pay paga */
  public void printCheck(String name, double pay) {
    outFile.println(name + " " + pay);
  }

  /** Mbyll output fajlin */
  public void close() {
    outFile.close();
  }
}
```

Përjashtimet dhe përpunuesit e përjashtimit

```
import javax.swing.*;
/** Pjesëton dy numra */
public class Division {
  public static void main(String[] args) {
    Division calc = new Division();
    int i = calc.readInt("Jepni një numër të plotë:");
    JOptionPane.showMessageDialog(null, "Herësi është " + 12 / i);
  }
  /** Lexon dhe kthen një numër të plotë
   * @param message mesazhi shfrytëzuesit
   * @return numri i lexuar */
  public int readInt(String message) {
    String s = JOptionPane.showInputDialog(message);
    int num = new Integer(s).intValue();
    return num;
  }
}
```

Hedhja e përjashtimeve

- Aplikacioni *hedh* një përjashtim të tipit `ArithmeticException` kur shfrytëzuesi jep 0 për input numër të plotë.
- Aplikacioni hedh një përjashtim të tipit `NumberFormatException` kur shfrytëzuesi jep një string jo numrorësh.
- Aplikacioni hedh një përjashtim të tipit `NumberFormatException` kur shfrytëzuesi trus një buton të dialogut para se të ketë radhitur stringun.

Përpunuesit e përjashtimit

- Sintaksa:

```
try { STATEMENTS }
catch( EXCEPTION ) {
  HANDLER
}
```

  ku *STATEMENTS* dhe *HANDLER* janë vargje urdhërash, *EXCEPTION* është klasë përjashtimesh, p.sh. `RuntimeException`.

- *HANDLER* quhet *përpunues i përjashtimit.*
- Semantika:
  - Fillon së ekzekutuari pjesa *STATEMENTS*.
  - Në qoftë se nuk hedhet përjashtim, atëherë klauza `catch` injorohet.
  - Por, në qoftë se hedhet përjashtim dhe është i tipit *EXCEPTION*, atëherë ekzekutohen urshërat *HANDLER* dhe aplikacioni vazhdon pa u terminuar: themi se përjashtimi është *kapur*.

Versioni i modifikuar i aplikacionit

```
public static void main(String[] args) {
  Division calc = new Division();
  try {
    int i = calc.readInt("Jepni një numër të plotë:");
    JOptionPane.showMessageDialog(null,
        "Herësi është " + 12 / i);
  } catch (ArithmeticException e) {
    JOptionPane.showMessageDialog(null,
        "Gabim në input ");
  } catch (NumberFormatException e) {}
}
```

Ristartimi i një metode me një përpunues përjashtimi

```
public int readInt(String message) {
  int num = 0;
  String s = JOptionPane.showInputDialog(message);
  try {
    num = new Integer(s).intValue();
  } catch (RuntimeException e) {
    if (s != null) {
      JOptionPane.showMessageDialog(null,
          s + " nuk është numër i plotë; provoni sërish!");
      num = readInt(message); //Ristarto me invokim rekursiv
    } else throw new NumberFormatException(e.getMessage());
  }
  return num;
}
```

Input interaktiv me përpunues përjashtimesh

`class DialogReader`	Lexon input interaktiv
`DialogReader()`	Konstrukton input view, një dialog
`String readString(String prompt)`	Lexon dhe kthen një string
`int readInt(String prompt)`	Lexon dhe kthen një int
`double readDouble(String prompt)`	Lexon dhe kthen një double

Tabela. API i klasës `DialogReader`

```
import java.text.*;
/** Konverton vlerën e temperaturës. */
public class FahrenheitToCelsius {
  public static void main(String[] args) {
    DialogReader reader = new DialogReader();
    int input = reader.readInt("Gradë Fahrenheit:");
    int f = new Integer(input).intValue();
    double c = (5.0 / 9.0) * (f - 32);
    DecimalFormat formatter = new DecimalFormat("0.0");
    System.out.println("Për gradë Fahrenheit " + f + ",");
    System.out.println("gradë Celsius " + formatter.format(c));
  }
}
```

Klasa DialogReader

```
import javax.swing.*;

/** Merr inputin e shfrytëzuesit nga një dialog */
public class DialogReader {
  /** Konstrukton input view, një dialog */
  public DialogReader() { }

  /** Lexon dhe kthen një string
   * @param prompt prompti i shfrytëzuesit
   * @return stringu që rradhit shfrytëzuesi */
  public String readString(String prompt) {
    return JOptionPane.showInputDialog(prompt);
  }
```

Klasa DialogReader (Vazhdim)

```
/** Lexon dhe kthen një int
 * @param prompt prompti i shfrytëzuesit
 * @return numri i plotë që rradhit shfrytëzuesi */
public int readInt(String prompt) {
  int answer = 0;
  String s = readString(prompt);
  try {
    answer = new Integer(s.trim()).intValue();
  } catch (RuntimeException e) {
    JOptionPane.showMessageDialog(null,
        "Gabim DialogReader: " + s + " jo int.");
    answer = readInt(prompt); // Ristarto
  }
  return answer;
}
```

Klasa DialogReader (Vazhdim)

```
  /** Lexon dhe kthen një double
   * @param prompt prompti i shfrytëzuesit
   * @return numri thyesor që rradhit shfrytëzuesi */
  public double readDouble(String prompt) {
    double answer = 0;
    String s = readString(prompt);
    try {
      answer = new Double(s.trim()).doubleValue();
    } catch (RuntimeException e) {
      JOptionPane.showMessageDialog(null,
          "DialogReader error: " + s + " not a double.");
      answer = readDouble(prompt); // restart
    }
    return answer;
  }
}
```

Përjashtimet janë objekte

- Kur të ndodhë një përjashtim (p.sh. pjesëtimi me zero), aplikacioni mbledh informacion mbi përjashtimin dhe ia dorëzon përpunuesit (*handler*) të përjashtimit:
  - përshkrimi i natyrës së përjashtimit (p.sh. „division by zero“);
  - përshkrimi ku ka ndodhur përjashtimi: rreshti i urdhërit, metoda, klasa dhe historia e invokimeve të metodave që kanë sjellur te përjashtimi.
- Këto copëza informacioni paketohen në një objekt përjashtimi të sapo konstruktuar: „rezultati“ i kompjutimit jokorrekt.
- Ky rezultat kërkon përpunuesin e vetë (themi se përjashtimi *hedhet*), që zakonisht është IDE ose JDK që ka startuar ekzekutimin e aplikacionit; përpunuesi afishon përmbajtjen e objektit në konsolë dhe aplikacioni terminohet.
- Programeri mund të shkruajë një përpunues përjashtimi, i cili *e kap* përjashtimin. Në këtë rast, ai obligohet ta korrigjojë kompjutimin jokorrekt në shkallë që ekzekutimi të vazhdojë në mënyrë të sigurtë. Pas përpunimit aplikacioni vazhdon me urdhërin vijues dhe evitohet terminimi.

Klasa Exception

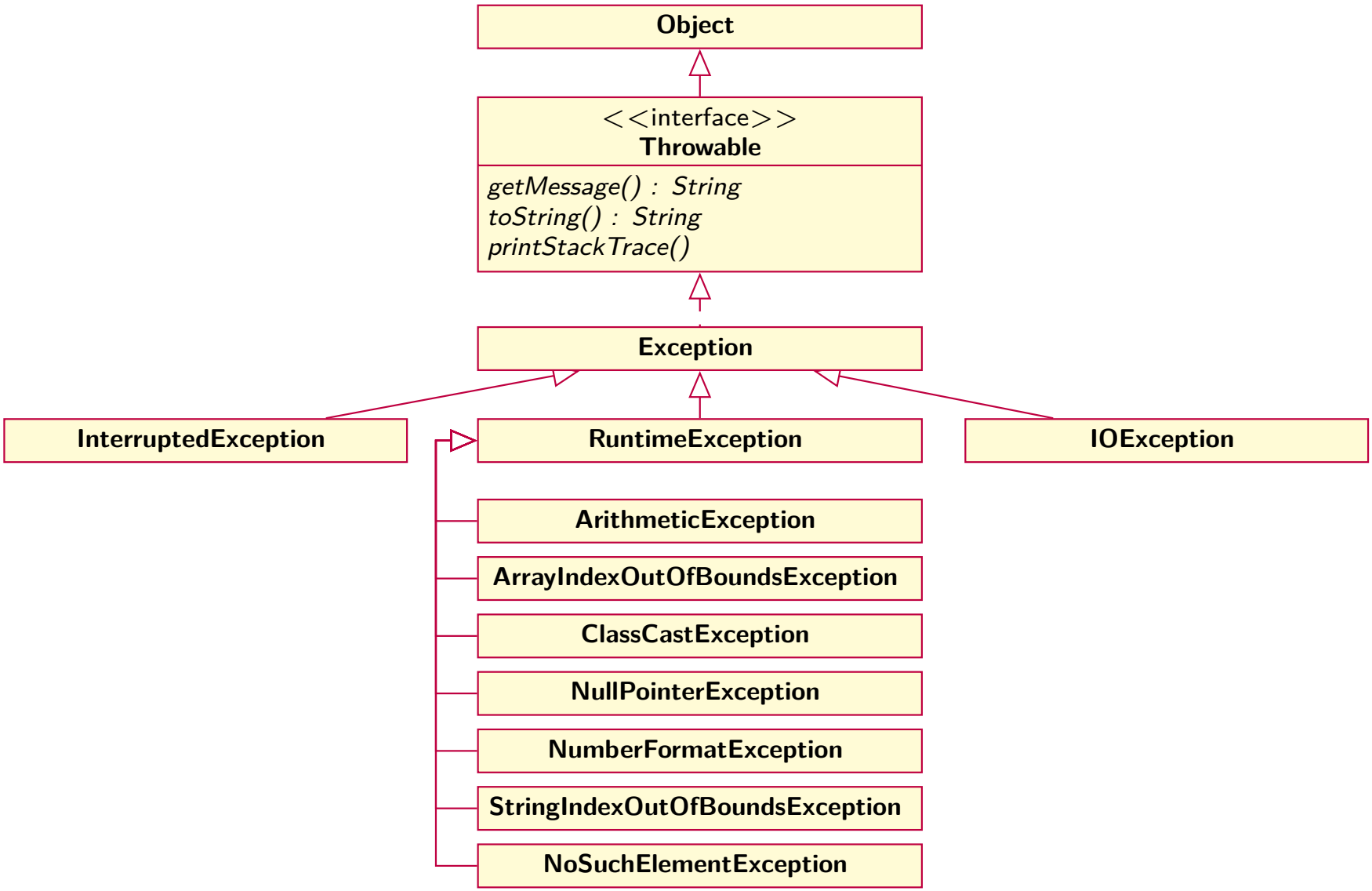
`class Exception`	Mbiklasa e përjashtimeve që mund të hedhen.
`String getMessage()`	Përshkruan përjashtimin
`String toString()`	Zakonisht, emri i klasës së përjashtimit dhe përshkrimi
`String printStackTrace()`	Shtyp në konsolë përshkrimin se ku ka ndodhur përjashtimi

Tabela. Disa metoda të klasës `Exception`

- Sintaksa:

```
try { STATEMENTS }
catch( EXCEPTION_TYPE e ) {
  HANDLER
}
```

Kierarkia e klasave të përjashtimeve



Shembull përpunimi gabimesh

```
public static void main(String[] args) throws IOException {
  Reader reader = new Reader();
  BufferedReader keyboard
      = new BufferedReader(new InputStreamReader(System.in));
  int i = reader.readIntFrom(keyboard);
  System.out.println(i);
}

public int readIntFrom(BufferedReader view) throws IOException {
  int num;
  System.out.print("Type an int: ");
  String s = view.readLine();
  try {
    num = new Integer(s).intValue();
  } catch (RuntimeException e) {
    System.out.println("Non-integer error");
    num = -1;
  }
  return num;
}
```

Versioni i modifikuar i shembulit

```
public static void main(String[] args) {
  Reader reader = new Reader();
  try {
    BufferedReader in = new BufferedReader(new FileReader("text.txt.in"));
    int i = reader.readIntFrom(in);
    System.out.println(i);
  } catch (IOException e) {
    System.out.println("Gabim: fajl joekzistent!");
  }
}

public int readIntFrom(BufferedReader view) {
  int num;
  System.out.print("Shtypni një int: ");
  try {
    String s = view.readLine();
    num = new Integer(s).intValue();
  } catch (RuntimeException e) {
    System.out.println("Gabim: jo int!");
    num = -1;
  } catch (IOException e) {
    System.out.println("Gabim: lexim fajli!");
    num = -1;
  }
  return num;
}
```

Përjashtimet e hedhura nga programeri

```
try {
    ... // ndërmerret ndonjë kompjutim kompleks
if ( PLOTËSOHET NDONJË KONDITË E KEQE )
{ throw new RuntimeException("gabim i tmerrshëm"); }
... // përndryshe, vazhdo me kompjutimin kompleks
}
catch (RuntimeException e) { // përpiqu të përpunosh konditën
  ... e.getMessage() ...
  if ( KONDITA NUK MUND TË PËRPUNOHET ME SUKSES ) {
    throw new RuntimeException("Abort!");
  }
}
```

Përfundim

- Në Java një output fajl më së thjeshti krijohet nga një `PrintWriter`:

```
PrintWriter outFile = new PrintWriter(new FileWriter("test.txt"));
```

  Objekti posedon metodat `print` dhe `println`.

- Një input fajl krijohet lehtë me një `BufferedReader`:

```
BufferedReader inFile = new BufferedReader(new FileReader("test.txt"));
```

  Objekti posedon metodën `readLine`.

- Input nga dritarja komanduese lexohet duke futur `System.in` në një `InputStreamReader`:

```
BufferedReader kbd
  = new BufferedReader(new InputStreamReader(System.in));
```

- Përjashtimet janë objekte, dhe programeri mund të shkruajë një përpunues përjashtimi (exception handler).