PROGRAMARE ORIENTATĂ PE OBIECTE

GENGE BÉLA

Capitolul 8
Java Swing. Canale de transfer date. Clase generice

... iarăși despre fire de execuție

- Fire daemon: fir de execuție ce permite terminarea aplicației de către JVM. Acestea rulează cât timp există fire utilizator.
- La lansarea în execuție se crează un fir de execuție user (non-daemon).
- JVM termină aplicația:
  - La apelul metodei exit() din clasa Runtime.
  - La terminarea tuturor firelor de execuție non-daemon.
  - Firele daemon sunt terminate automat de JVM dacă nu mai rulează fire user.

Crearea firelor daemon

- Se crează (se instanțiază clasa/clasele) un fir de execuție user prin una din cele două metode studiate:
  - Moștenire clasa Thread.
  - Implementare interfața Runnable.
- Se apelează metoda setDaemon(true) din clasa Thread pentru a modifica tipul firului.
- Se apelează metoda start() pentru lansarea firului de execuție.
- Atenție! Apelul setDaemon(true) după apelul start() va arunca o excepția:
  - IllegalThreadStateException.

Java Swing

- Un pachet pentru implementarea interfețelor grafice: javax.swing.
- Toate obiectele grafice dintr-o aplicație sunt administrate de către firul de execuție: Event Dispatcher Thread (EDT).
- EDT este altul decât firul principal (care execută metoda main).
- EDT este un fir non-daemon.
  - Terminarea aplicației necesită terminarea firului EDT (dar nu deținem controlul asupra acestui fir!).
  - Soluția: apel Exit: frame.setDefaultCloseOperation(**JFrame**.EXIT_ON_CLOSE);

Crearea aplicațiilor cu GUI Java

- Abordarea naivă:

```
public class Exemplu {
    public static void main(String args[]) {
        // Exemplu de cod sursa: https://en.wikipedia.org/wiki/Swing_(Java)
        JFrame f = new JFrame("Hello!");

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        f.setLayout(new FlowLayout());

        f.add(new JLabel("Hello, world!"));

        f.add(new JButton("Press me!"));

        f.pack();

        f.setVisible(true);
    }
}
```

Crearea aplicațiilor cu GUI Java

- Problema:
  - Crearea obiectelor grafice și adăugarea lor pe firul principal.
  - Împrospătarea obiectelor GUI pe EDT.
  - → race condition, posibile inconsistențe, operația nu este thread-safe!

Crearea aplicațiilor cu GUI Java

- Soluția: toate operațiile de accesare a GUI să fie efectuate de EDT.
- Mecanismul: comunicarea cu EDT.
- Mai exact:
  - Apelul metodei SwingUtilities.invokeLater(Runnable).
- invokeLater() este o metodă ce depune un nou “mesaj” în coada de mesaje a EDT.
- Mesajul adăugat conține o referință către un obiect ce implementează interfața Runnable.
- Operația de accesare a GUI se realizează prin apelul metodei run (de către EDT).

Crearea aplicațiilor cu GUI Java

- Exemplu de implementare a soluției:

```java
public class SwingExample implements Runnable {

    @Override
    public void run() {
        // Create the window
        JFrame f = new JFrame("Hello, !");
        // Sets the behavior for when the window is closed
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Add a layout manager so that the button is not placed on top of the label
        f.setLayout(new FlowLayout());
        // Add a label and a button
        f.add(new JLabel("Hello, world!"));
        f.add(new JButton("Press me!"));
        // Arrange the components inside the window
        f.pack();
        // By default, the window is not visible. Make it visible.
        f.setVisible(true);
    }

    public static void main(String[] args) {
        SwingExample se = new SwingExample();
        // Schedules the application to be run at the correct time in the event queue.
        SwingUtilities.invokeLater(se);
    }

}
```

- Sursa: https://en.wikipedia.org/wiki/Swing_(Java)

Crearea aplicațiilor cu GUI Java

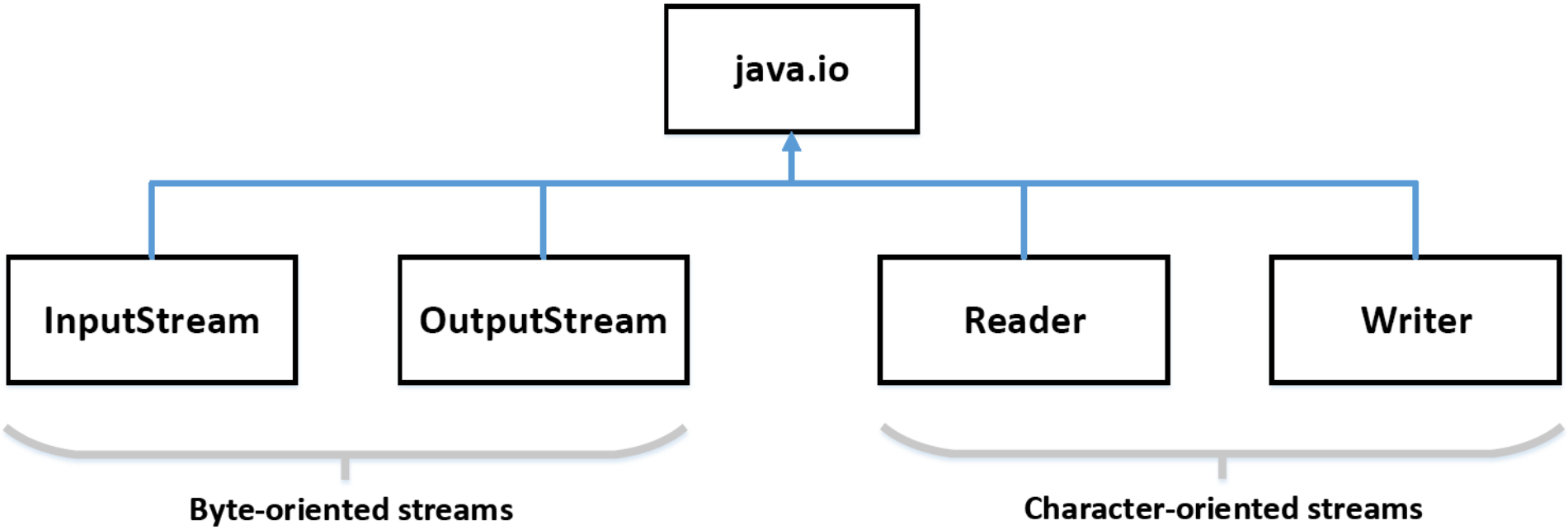
- Alte metode din SwingUtilities:
  - isEventDispatchThread(): interogare tip fir (EDT sau normal).
  - invokeAndWait(Runnable): blochează firul curent până la terminarea execuției metodei run.

Canale de transfer de date

- Canale de transfer: citire scriere.
- Canalele asigură:
  - Citire/scriere octet.
  - Citire/scriere caractere unicode (2-4 octeți).
  - Citire/scriere șiruri de caractere.
  - Citire/scriere tipuri de date (int, byte, ...).
  - Citire/scriere obiecte.
- Metodele specifice în pachetul: java.io.
- Canale standard:
  - System.in.
  - System.out.
  - System.err.

Canale de transfer de date

- Ierarhia de clase principală din java.io:



Canale bazate pe octeți

- FileOutputStream.
- FileInputStream.

- Metode uzuale:
  - write(byte);
  - write(byte[]);

Canale bazate pe caractere

- FileWriter.
- FileReader.

- Metode uzuale:
  - write(char);
  - write(String);

Clase generice

- O clasă generică este parametrizată peste tipurile de date încapsulate.
- Posibilă soluție:
  - Clasă care stochează un obiect de tipul Object și include metode get/set.

```
 3 public class Exemplu {
 4
 5     private Object obj;
 6
 7     public Object get() {return obj;}
 8     public void set(Object obj) {
 9         this.obj = obj;
10     }
11 }
```

Problema

- Problema cu exemplul anterior:
  - Clasa Exemplu permite apelul metodei “set” pe aceeași instanță cu tipuri diferite.
  - Astfel:

```
 3 public class Exemplu {
 4
 5     private Object obj;
 6
 7     public Object get() {return obj;}
 8     public void set(Object obj) {
 9         this.obj = obj;
10     }
11 }
12
13 Exemplu ex = new Exemplu();
14 ex.set(new Persoana());
15 ex.set(new Calculator());
16 ((Persoana)ex.get()).setNume("Popescu");
```

Soluția

- Înlocuirea tipului asupra căruia se aplică clasa **în timpul compilării**.
- Clasa definește un **placeholder** care **va fi înlocuit la compilare** cu tipul dorit.
- Pentru exemplul anterior:

```
13 public class Exemplu<T> {
14
15     private T obj;
16
17     public T get() {return obj;}
18     public void set(T obj) {
19         this.obj = obj;
20     }
21 }
22 Exemplu<Persoana> ex = new Exemplu<Persoana>();
23 ex.set(new Persoana());
24 ex.set(new Calculator());//!EROARE DE COMPILARE
```

Valoarea lui “T”

- “T” reprezintă o convenție de notare din Java. Acesta denotă un tip anume.
- În locul lui “T” se poate scrie orice literă – se recomandă menținerea convenției pentru evitarea confuziilor.
- Alte litere uzuale:
  - K – Key (în construcții Key-Value).
  - V – Value (în construcții Key-Value).
  - N – Number.
  - T – Type.
  - S, U, V – alte tipuri.
- Valoarea lui T poate fi orice tip **non-primitiv**!

Tipuri primitive

- Tipuri de date primitive în Java:
  - byte.
  - short.
  - int.
  - long.
  - float.
  - double.
  - boolean.
  - char.
- Tipurile primitive NU pot înlocui T!

Exemplu

```
32 public class Exemplu<K,V> {
33
34     private K key;
35     private V value;
36
37     public K getKey() {return key;}
38     public V getValue() {return value;}
39
40     public void set(K key, V value) {
41         this.key = key;
42         this.value = value;
43     }
44 }
45
46 // EROARE COMPILARE: int este un tip de data primitiv.
47 Exemplu<int,String> ex = new Exemplu<int,String>();
48
49 // OK: Integer incapsuleaza un tip de data intreg primitiv.
50 Exemplu<Integer,String> ex = new Exemplu<Integer,String>();
```

Probleme

- Să se implementeze stiva statică folosind tipuri generice.

HashMap

- Asigură un mecanism pentru accesarea rapidă a unor obiecte, pe baza conceptului cheie-valoare.
- Metode utile:
  - Object.hashCode(): int.
  - Începând cu Java 7: Objects.hash(Object ... Values): int.
- Exemplu de suprascriere a metodei hashCode pentru clasa Persoana.
  - Varianta1: implementare customizată complet.
  - Varianta2: implementare cu apelul metodei hashCode din String.
  - Varianta3: implementare cu apelul Objects.hash.

Problemă

- Să se implementeze CustomHashMap, luând în considerare:
  - Tipuri generice pentru cheie (K) și valoare (V).
  - Metoda int hashCode() din Object.
  - Mai multe obiecte pot duce la aceeași cheie.
  - Metode specifice:
    - void put(K key, V value)
    - V get(K key)
    - boolean exists(K key)