Curs 14 POO
© Crenguta M. Puchianu

- Interfete functionale predefinite
- Lucrul cu fluxuri de elemente

Interfete functionale predefinite

In pachetul java.util.function se afla 4 interfeţe funcţionale predefinite: Predicate, Consumer, Function şi Supplier.

```
public interface Predicate<T>{
 boolean test(T t);
}
```

Operaţia test() verifică dacă condiţia t este îndeplinită şi returnează true sau false.

```
public interface Consumer<T>{
 void accept(T t);
}

public interface Function<T,R>{
 R apply(T t);
}
```

Operaţia apply() primeşte un argument de tip t şi returnează un rezultat de tip R.

```
public interface Supplier<T>{
 T get();
}
```

Operaţia get() returnează o valoare apelantului.

Lucrul cu fluxuri de elemente. Surse pentru fluxuri

Flux = secventa de elemente: valori de tip primitiv de date sau referinte la obiectele unei clase.

Un flux are un inceput (o sursa), mijloc (operatii intermediare) şi un sfârşit (o operatie terminală).

O interfata utila este **Stream** din pachetul java.util.stream.

Surse pentru fluxuri:

1. Interfaţa IntStream are metoda range(int n1, int n2) ce genereaza un flux de numere întregi din intervalul [n1, n2]. Exemplu: `IntStream.range(1,6)`

2. Interfaţa IntStream are metoda iterate(int n1, Function f) care ia numarul n1 şi aplica iterativ functia f obtinând un şir infinit de elemente întregi. El trebuie să fie restrictionat cu metoda limit(int n). IntStream.iterate(1, i->i*2).limit(5)

3. Clasa utilitara Arrays are metoda statica stream() a produce un flux de elemente sau referinte la obiectele tabloului dat ca parametru. Arrays.stream(new int[]{1,2,3,5,6})

4. Interfaţa Stream (din java.util.stream) are metoda statică of(elemente separate prin virgula) creaza un flux cu elementele transmise ca parametri. Stream.of(1,2,3,5,6)

5. Interfaţa Stream are metoda statică generate(Supplier<T> s) care genereaza un flux infinit de elemente obtinute cu s.Stream.generate(()->25).limit(3)

Lucrul cu fluxuri de elemente. Surse pentru fluxuri (cont.)

6. Interfata Collection are metoda stream() ce furnizeaza un flux format din elementele colectiei pe care este apelată metoda.

7. Clasa java.nio.file.Files conţine metoda lines() un flux de linii ce formeaza continutul fişierului transmis ca parametru.

8. Clasa java.util.Pattern are metoda splitAsStream(String s) returnează un flux de subşiruri ale lui s după un anumit pattern.

Pattern.compile(“ “).splitAsStream(“a cb drds”)

6. Clasa java.util.Random are metoda ints() ce returnează un flux infinit de numere intregi aleatoare. El trebuie sa fie restrictionat cu metoda limit(int n).

7. Clasa String moşteneşte de la clasa CharSequence metoda chars() ce returneaza un flux de numere intregi (de tip IntStream) cu codurile Unicode ale caracterelor şirului transmis ca parametru.

Operatii intermediare pe fluxuri

1. Stream<T> filter(Predicate<? Super T> conditie) returnează un flux cu elementele fluxului initial care verifica predicatul conditie.

2. Stream<R> map(Function<? Super T, ? extends R> f) aplica functia f fiecarui element al fluxului initial şi returneaza noul flux.

3. Stream<T> distinct() – returnează un flux din fluxul de baza în care a eliminat duplicatele (pe baza metoda equals())

4. Stream<T> sorted() – returnează un flux ce contine elementele sortate in ordine naturala a fluxului de baza

5. Stream<T> sorted(Comparator <? super T> compare) – returnează un flux ce contine elementele sortate conform lui compare a fluxului de baza

5. Stream<T> limit(long size) - returnează un flux cu size numar de elemente

Operatii terminale pe fluxuri

1. void forEach(Consumer<? super T> actiune) aplica actiunea pe fiecare element al fluxului curent

2. Optional<T> min(Comparator<? super T> compare) returneaza elementul minim al fluxului curent

3. Optional<T> max(Comparator<? super T> compare) returneaza elementul maxim al fluxului curent

4. Long count() – returneaza numarul de elemente ale fluxului curent

5. <R,A> R collect(Collector<? super T,A,R> collector) care memorează elementele fluxului curent într-o colecție dată ca parametru.

Operatii terminale pe fluxuri (cont.)

6. boolean allMatch(Predicate<? super T> p) verifică dacă toate elementele fluxului curent verifică predicatul p.
7. boolean anyMatch(Predicate<? super T> p) verifică dacă cel puțin un element al fluxului curent verifică predicatul p.
8. boolean noneMatch(Predicate<? super T> p) verifică dacă niciun element al fluxului curent nu verifică predicatul p.

```
boolean p=IntStream.of(1,2,3,5).noneMatch(p -> p==4);
```

9. Object[] toArray() – returnează un tablou de obiecte ale clasei Object cu elementele fluxului