Programare multiparadigmă - **JAVA**

Prof. univ. dr. **Claudiu Vințe**

claudiu.vinte@ie.ase.ro

Java *Generics*

- Mecanismul permite parametrizarea tipurilor utilizate la definirea de clase, interfețe sau metode

- Concept similar celui de *template* din C++ sau *generics* din C#

- Mecanism de implementare – eradicarea (ștergererea) tipurilor
  - ștergerea tipului este un proces în care compilatorul înlocuiește un parametru generic cu o clasă reală sau cu o metodă de tip *bridge (punte)*
  - înlocuiește parametrii de tip în tipul generic cu limita lor, dacă sunt utilizați parametrii de tip mărginit
  - înlocuiește parametrii de tip în tipul generic cu *Object*, dacă sunt utilizați parametrii de tip nelimitat
  - introducere conversie de tip (*casting*) pentru a păstra siguranța tipului
  - generare metode de punte pentru a păstra polimorfismul în tipurile generice extinse

- Utilizat în special pentru definirea colecțiilor:
  - eliminare necesitate cast la extragerea elementelor
  - eliminare posibilitate de adăugare elemente de tipuri diferite în colecții

Java *Generics*

- Declarare clase / interfețe generice:

  [modificatori] class NumeClasa<T1, T2, ..., Tn> { ... }

- Declarare metode generice:

  [modificatori] <T1, T2, ..., Tn> tip numeMetoda(listaParametri) [throws E1,...,Em] {...}

- Raw type – tipul obținut prin procesul de eradicare a tipului din tipul parametrizat

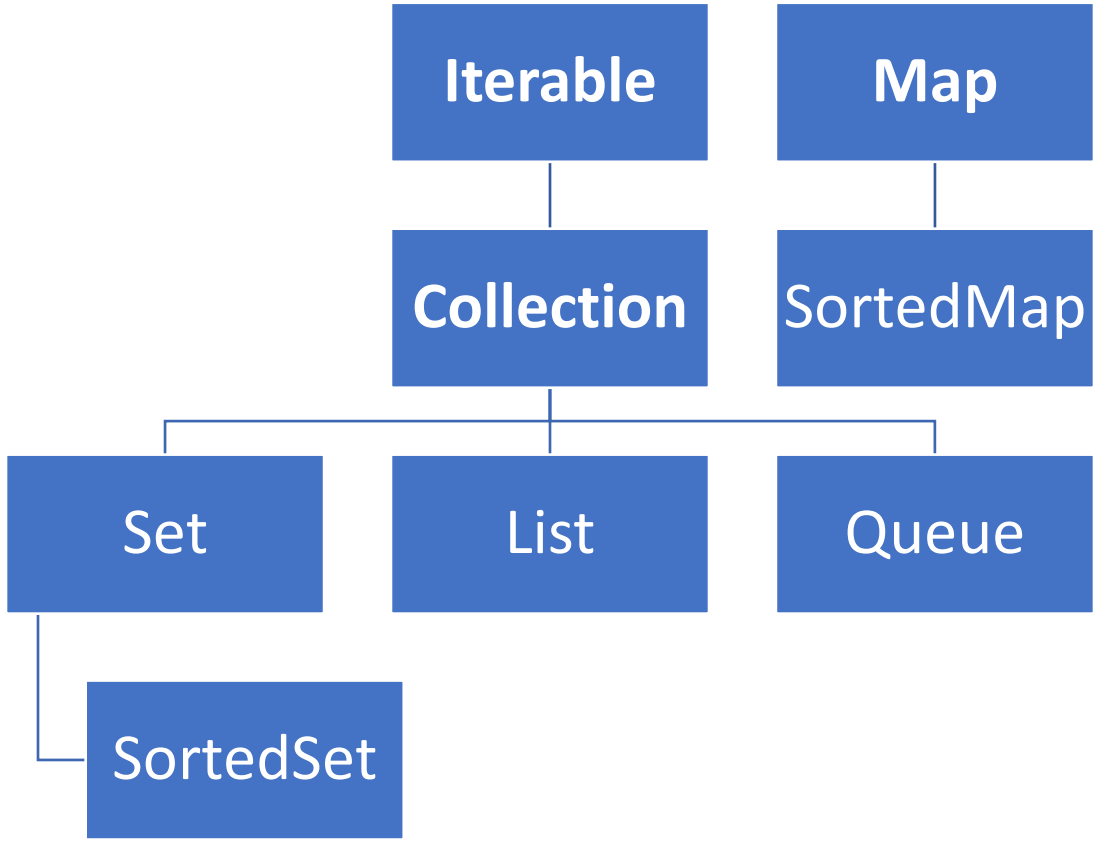
```
class Test<T> {
    public void metoda(Object element) {
        if (element instanceof T) {  // eroare compilare
            // ...
        }
        T item2 = new T();          // eroare compilare
        T[] iArray = new T[10];     // eroare compilare
        T obj = (T)new Object();    // warning la compilare
    }
}
```

Java Collection Framework - JCF

- Colecție = un obiect care grupează mai multe obiecte
- JCF este compusă din:
  - Interfețe
    - Tipuri abstracte care reprezintă colecțiile
    - Permit utilizarea obiectelor colecție independent de modalitatea de implementare
    - Modul preferat de utilizare a colecțiilor
  - Implementări
    - Clase care furnizează implementări concrete ale interfețelor JCF
  - Algoritmi
    - Metode care furnizează implementări polimorfice pentru diverse operații uzuale (sortare, căutare)
    - Lucrează pe baza interfețelor JCF
- Toate componentele JCF sunt generice

JCF – Interfețele de bază

- **Collection** – reprezintă un grup de obiecte (elemente) și furnizează metodele de bază pentru enumerarea, adăugarea, ștergerea și căutarea elementelor
- Colecții:
  - **Set** – nu poate conține valori duplicate
  - **List** – poate conține valori duplicate și permite accesul direct la elemente
  - **Queue** – utilizate pentru a organiza elemenetele în vederea prelucrării conform unei strategii (FIFO, prioritate, etc.)
- **Map** – reprezintă o colecție de perechi cheie (unică) – valoare și furnizează metodele de bază pentru citirea și manipularea perechilor
- Determinarea egalității se realizează pe baza metodei ***Object.equals***, iar ordonarea pe baza implementării interfeței ***Comparable<T>*** *sau* ***Comparator<T>***



JCF – Implementări de bază

Interfața	*Hash Table*	*Vectori*	*Arbori*	*Liste*	*Hash + Listă*
`Set`	***`HashSet`***		`TreeSet`		`LinkedHashSet`
`List`		***`ArrayList`***		`LinkedList`	
`Queue`					
`Map`	***`HashMap`***		`TreeMap`		`LinkedHashMap`

Reprezintă clasele recomandate pentru majoritatea aplicațiilor

Nu sunt clase abstracte – furnizează implementarea completă pentru interfețele respective

Permit elemente ***`null`*** (atât chei cât și valori)

JCF - Algoritmi

- Implementați sub formă de metode statice în clasa *java.util.**Collections***
- Aranjare elemente
  - ***sort*** – sortare simplă sau pe bază de obiect comparator
  - ***shuffle*** – amestecare elemente
  - ***reverse*** – inversarea ordinii elementelor
- Căutare
  - ***binarySearch*** – căutare în liste sortate
  - ***frequency*** – numărare apariții element
- Manipulări de bază
  - ***addAll*** - adăugare elemente multiple
  - ***fill*** – umplere lista cu o anumită valoare
  - ***copy*** – copiere elemente între două liste
  - ***min / max*** – determinare elemente minime și maxime