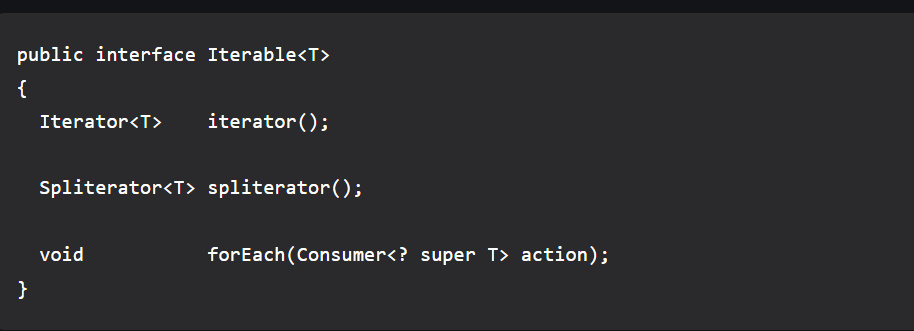


* **Collection** – orice grup individual de obiecte care sunt reprezentate ca o unitate
* Interfata de baza la toate clasele si interfatele din framework este **Iterable**

**Iterable**

* Rolul la aceasta interfata este de a oferi un iterator in cazul oricarei structuri de date
* Are 3 metode abstracte:

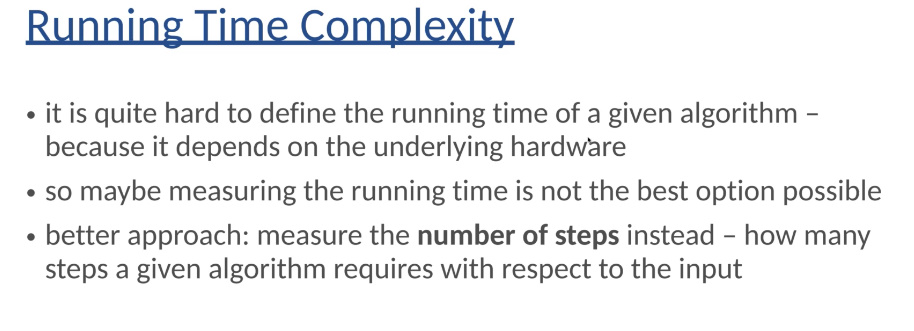


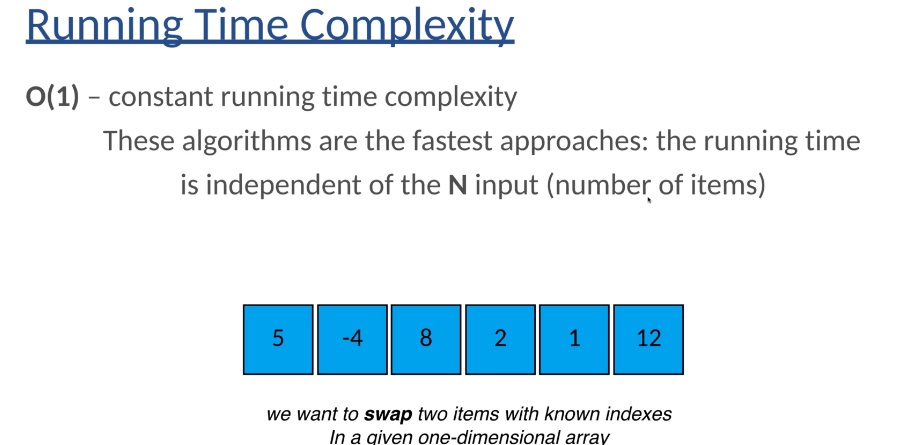
* Clasa care implementeaza iterable poate fi iterata, desi trebuie sa o facem noi evident.
* Un obiect care e si de tip Iterable pote fi folosit intr-un foreach loop
* Collection interface ofera metodeel:
* add(Object)
* addAll(Collection)
* clear()
* contains(Object)
* containsAll(Collection)
* equals(Object)
* hashCode()
* isEmpty()
* iterator()
* parallelStream()
* remove(Object)
* removeAll(Collection)
* removeIf(Predicate)
* retainAll(Collection)
* size()
* stream()
* spliterator()
* toArray()

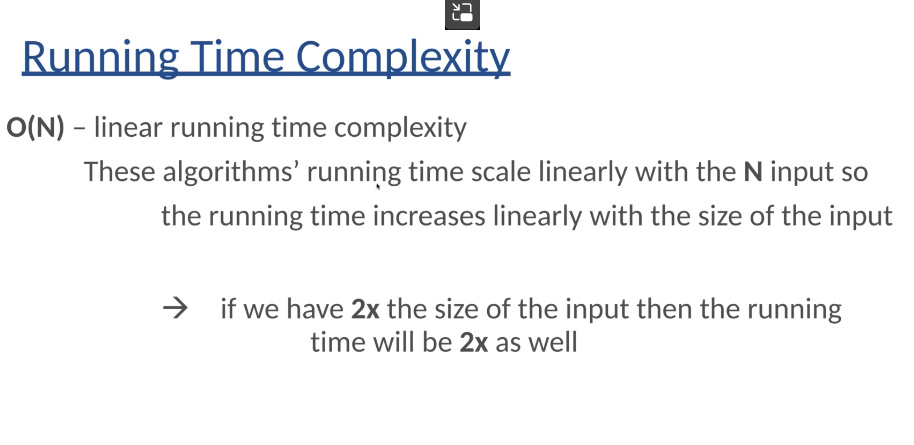
**Iterator**

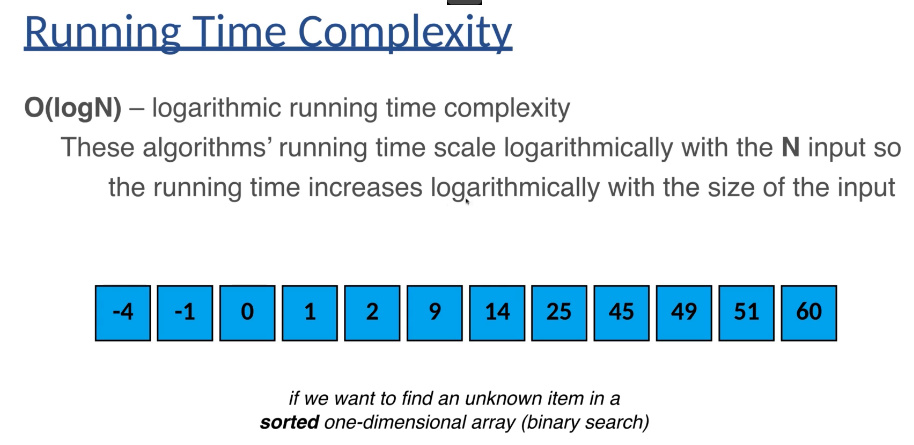
* Iterator este o clasa care ne ajuta sa cream obiecte ce stocheaza toate elementele unei structuri de date, sau clase create de noi
* Orice structura de date are metoda .iterator() care returneaza un iterator cu toate itemele sale
* Metode:
* hasNext() - returneaza true daca mai are elemente si false daca nu
* next() – returneaza urmatorul element
* remove() – sterge elementul curent din structura de date, nu doar din iterator. Metoda poate fi apelata doar odata per next()
* public class Test {  
   public static void main(String[] args) {  
   List<Integer> list = Arrays.*asList*(1,2,3,4,5,6);  
   Iterator<Integer> iterator = list.iterator();  
   while(iterator.hasNext())  
   System.*out*.println(iterator.next());  
   }  
  }

**Complexity**









**Data Structures**

