Introduction to collections:

A *collection* — sometimes called a container — is simply an object that groups multiple elements into a single unit. Collections are used to store, retrieve, manipulate, and communicate aggregate data. Typically, they represent data items that form a natural group, such as a poker hand (a collection of cards), a mail folder (a collection of letters), or a telephone directory (a mapping of names to phone numbers).

Collections Framework:

A *collections framework* is a unified architecture for representing and manipulating collections. All collections frameworks contain the following:

* **Interfaces:** These are abstract data types that represent collections. Interfaces allow collections to be manipulated independently of the details of their representation. In object-oriented languages, interfaces generally form a hierarchy.
* **Implementations:** These are the concrete implementations of the collection interfaces. In essence, they are reusable data structures.

**Algorithms:** These are the methods that perform useful computations, such as searching and sorting, on objects that implement collection interfaces. The algorithms are said to be *polymorphic*: that is, the same method can be used on many different implementations of the appropriate collection interface. In essence, algorithms are reusable functionality.

The list of methods belonging to the **Collection** interface is shown in the table given below

|  |  |
| --- | --- |
| **Method** | **Uses** |
| add(Element) | Adds the specified element to this set if it is not already present (optional operation). |
| clear() | Removes all of the elements from this set (optional operation). |
| contains(Object o) | Returns true if this set contains the specified element. |
| equals(Object o) | Compares the specified object with this set for equality. |
| hashCode() | Returns the hash code value for this set. |
| isEmpty() | Returns true if this set contains no elements. |
| iterator() | Returns an iterator over the elements in this set. |
| remove(Object o) | Removes the specified element from this set if it is present (optional operation). |
| size() | Returns the number of elements in this set (its cardinality). |

w.a.p to store a group of objects into an array and retrieve the

object data and display

\*/

import java.util.Scanner;

class Employee

{

int age;

String name;

public Employee(String name, int age)

{

this.age=age;

this.name=name;

}

public void display()

{

System.out.println(name + "\t" + age);

}

}

public class EmpTest

{

public static void main(String args[])

{

Scanner in = new Scanner(System.in);

int age;

String na ;

Employee e[] = new Employee[5]; // e is object array

for(int i=0; i<5; i++) {

System.out.println("enter name, age");

na=in.next();

age=in.nextInt();

e[i] = new Employee(na, age);

}

for(Employee t : e) { //for(int i=0; i<5; i++)

t.display(); // e[i].display();

}

}

}