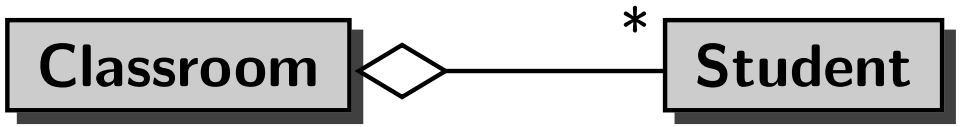
Collection



A `Classroom` has a collection of `Students`.

List

Question

May a list contain duplicates?

List

Question

May a list contain duplicates?

Answer

Yes.

List

Question

May a list contain duplicates?

Answer

Yes.

Question

Are the elements of a list ordered?

List

Question

May a list contain duplicates?

Answer

Yes.

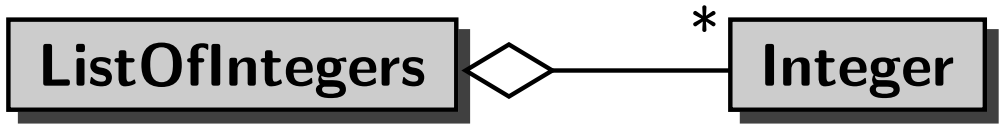
Question

Are the elements of a list ordered?

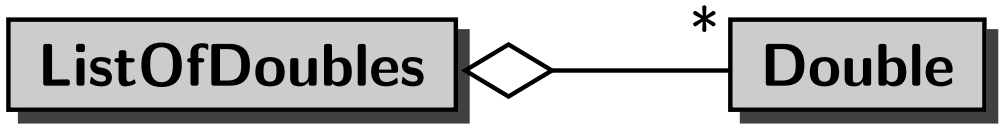
Answer

Yes.

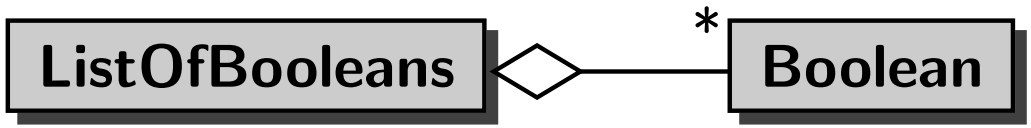
Number of students attending lectures



Score for each test



Attendance of a student



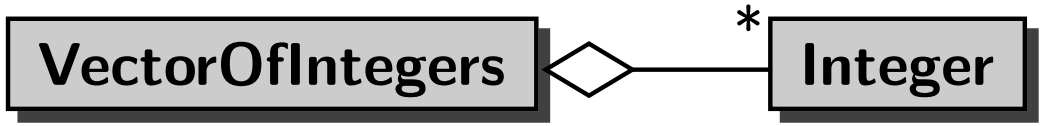
Number of students attending lectures

The list is implemented by means of an array.

Number of students attending lectures

The list is implemented by means of a "links."

Number of students attending lectures



The list is implemented by means of an array and multiple threads can manipulate the list at the same time.

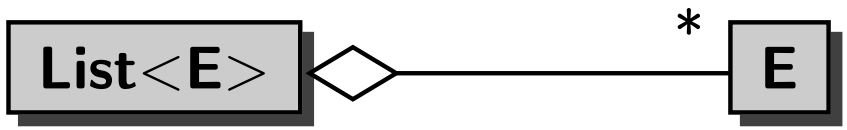
Lists

These different lists can be classified based on

- the type of the elements of the list (Integer, Double, Boolean, ...) and
- the way the list is implemented (using an array, using "links," ...).

Lists

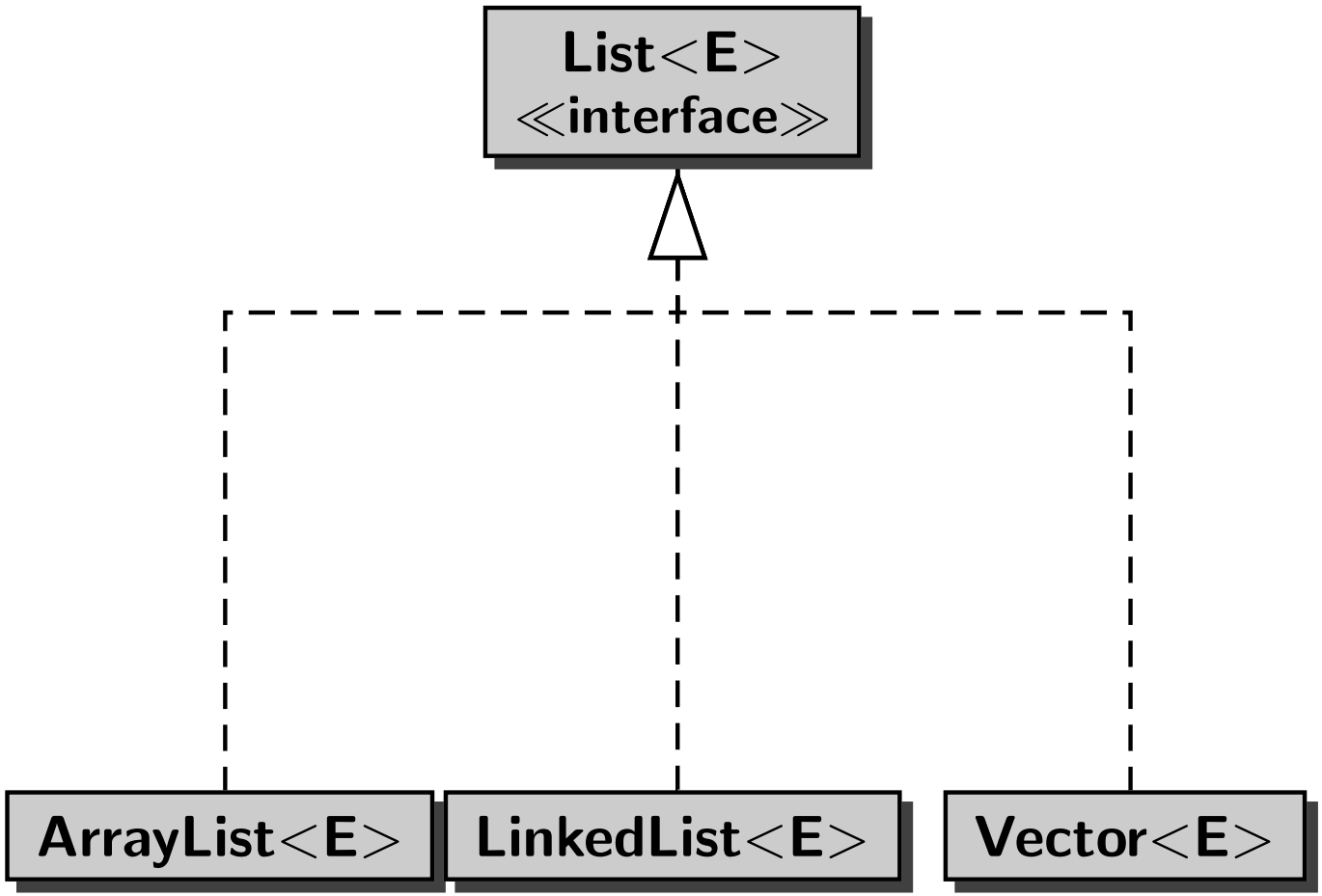
To abstract from the type of the elements of the list, we exploit generics.



E is a type parameter. The elements of the list are of type E.

Lists

To abstract from the way the list is implemented, we exploit interfaces.



Class versus Interface

interface	specification	what?
class	implementation	how?

Number of students attending lectures

```
final int LECTURES = 24;
List<Integer> attendance =
    new ArrayList<Integer>(LECTURES);
```

- The type of the elements is `Integer` and
- the list is implemented by means of an array.

Number of students attending lectures

```
final int LECTURES = 24;
List<Integer> attendance =
    new ArrayList<Integer>(LECTURES);
```

- The type of the elements is `Integer` and
- the list is implemented by means of an array.

Question

Why can we assign an object of type `ArrayList<Integer>` to a variable of type `List<Integer>`?

Number of students attending lectures

```
final int LECTURES = 24;
List<Integer> attendance =
    new ArrayList<Integer>(LECTURES);
```

- The type of the elements is `Integer` and
- the list is implemented by means of an array.

Question

Why can we assign an object of type `ArrayList<Integer>` to a variable of type `List<Integer>`?

Answer

Because the class `ArrayList<E>` implements the interface `List<E>`.

Score for each test

```
List<Double> tests = new LinkedList<Double>();
```

- The type of the elements is `Double` and
- the list is implemented by means of "links."

ArrayList, LinkedList or Vector?

Depends on which operations on the list are performed.

Question

How many milliseconds does it take to add n elements to the end of a list?

ArrayList, LinkedList or Vector?

Depends on which operations on the list are performed.

Question

How many milliseconds does it take to add n elements to the end of a list?

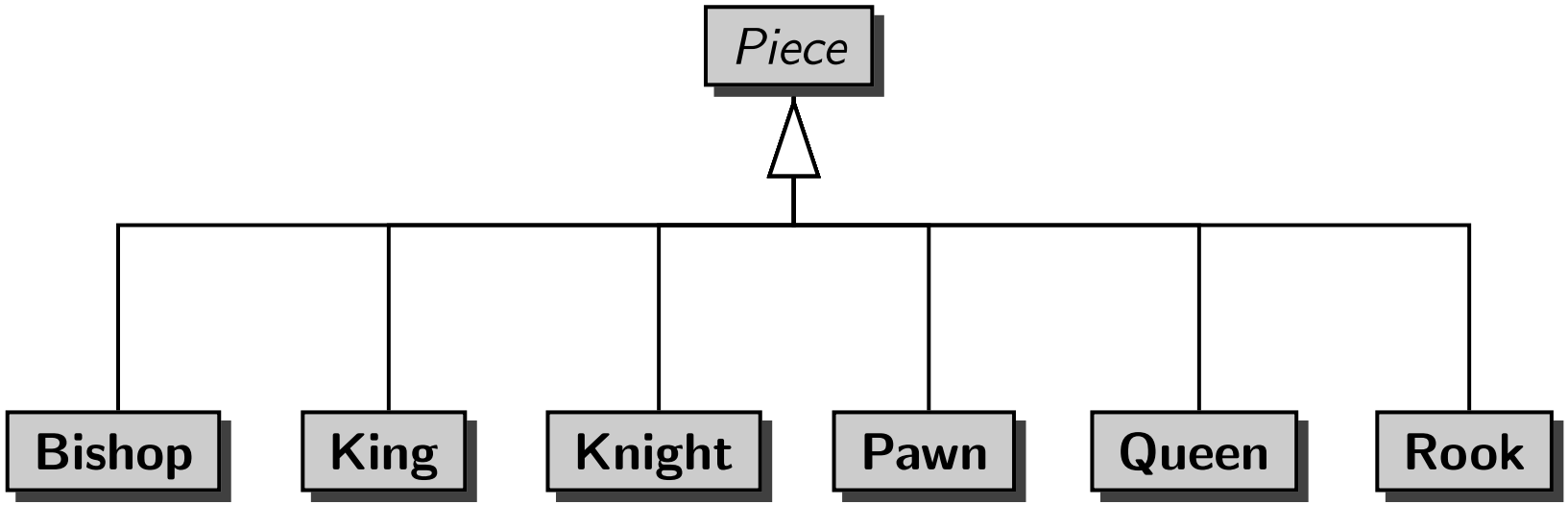
Answer

n	ArrayList	LinkedList	Vector
10^5	9	12	14
10^6	47	92	113
10^7	442	824	1041
2×10^7	913	1,650	2,076
3×10^7	1,350	143,616	3,230
4×10^7	2,527		4,103
5×10^7	2,689		6,119

ArrayList, LinkedList or Vector?

- Adding to or deleting from the beginning of a `LinkedList` is in general more efficient than adding to or deleting from the beginning of an `ArrayList` or `Vector`.
- Adding and deleting while traversing a `LinkedList` is in general more efficient than adding and deleting while traversing an `ArrayList` or `Vector`.
- In most other cases, `ArrayList` outperforms `LinkedList` and `Vector`.

Chess pieces



Row of a chess board

Question

How do you represent a row of a chess board?

Row of a chess board

Question

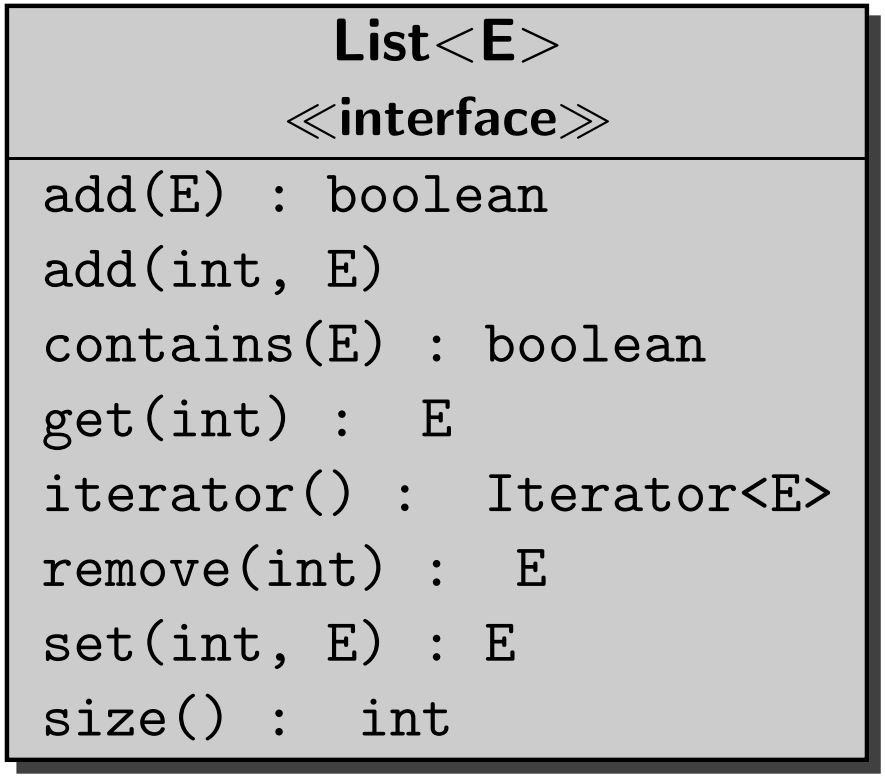
How do you represent a row of a chess board?

Answer

```
final int COLUMNS = 8;
List<Piece> row = new ArrayList<Piece>(COLUMNS);
```

- The type of the elements is `Piece` and
- the list is implemented by means of an array.

Methods of List



Row of a Chess board

Question

Create an empty row of a chess board.

Row of a Chess board

Answer

```
final int COLUMNS = 8;
List<Piece> row = new ArrayList<Piece>(COLUMNS);
for (int c = 0; c < COLUMNS; c++) {
   row.add(null);
}
```

Row of a Chess board

Question

Place a black rook on the first and the last square of the row.

Row of a Chess board

Answer

```
Rook rook = new Rook(Color.BLACK);
row.set(0, rook);
row.set(COLUMNS - 1, rook);
```

Row of a Chess board

Question

Place a white pawn on each square of the row.

Row of a Chess board

Answer

```
Pawn pawn = new Pawn(Color.WHITE);
for (int c = 0; c < COLUMNS; c++) {
   row.set(c, pawn);
}
```

Row of a Chess board

Question

Print the row.

An empty square is represented by two spaces. A non-empty square is represented by the representation of the piece on that square. For example, a black king is represented by BK and a white queen is represented by WQ.

The squares are separated by a single space.

Row of a Chess board

Answer

```
StringBuffer representation = new StringBuffer();
for (Piece piece : row) {
   if (piece == null) {
      representation.append("  ");
   } else {
      representation.append(piece.toString());
   }
   representation.append(" ");
}
output.println(representation.toString());
```

Question

May a set contain duplicates?

Sets

Question

May a set contain duplicates?

Answer

No.

Sets

Question

May a set contain duplicates?

Answer

No.

Question

Are the elements of a set ordered?

Question

May a set contain duplicates?

Answer

No.

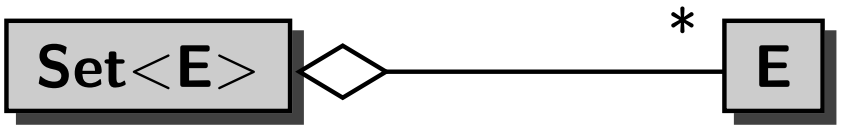
Question

Are the elements of a set ordered?

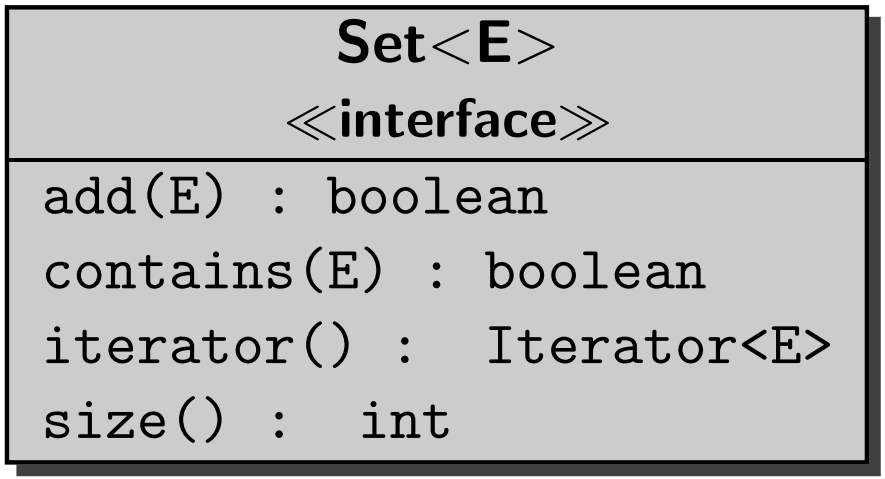
Answer

No.

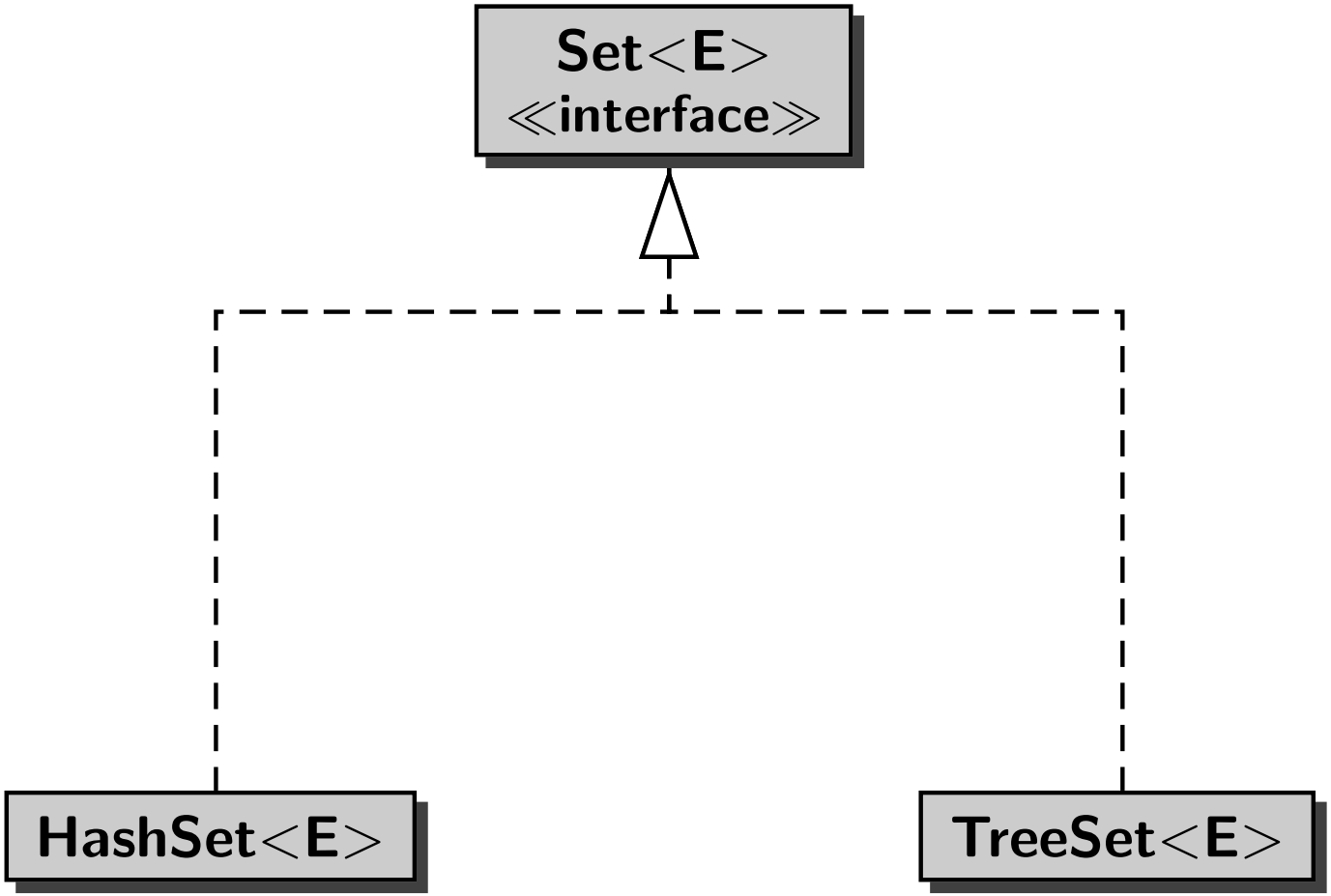
Sets



Methods of Set



Sets



HashSet or TreeSet?

- Adding to or deleting from or searching in a `HashSet` is in general more efficient than adding to or deleting from or searching in a `TreeSet`.
- `TreeSet` keeps the elements sorted, but `HashSet` does not.

ITunes library

Problem

Print each song of each playlist of an iTunes library. Each song should appear on a separate line. Playlists should be separated by a blank line.

ITunes library

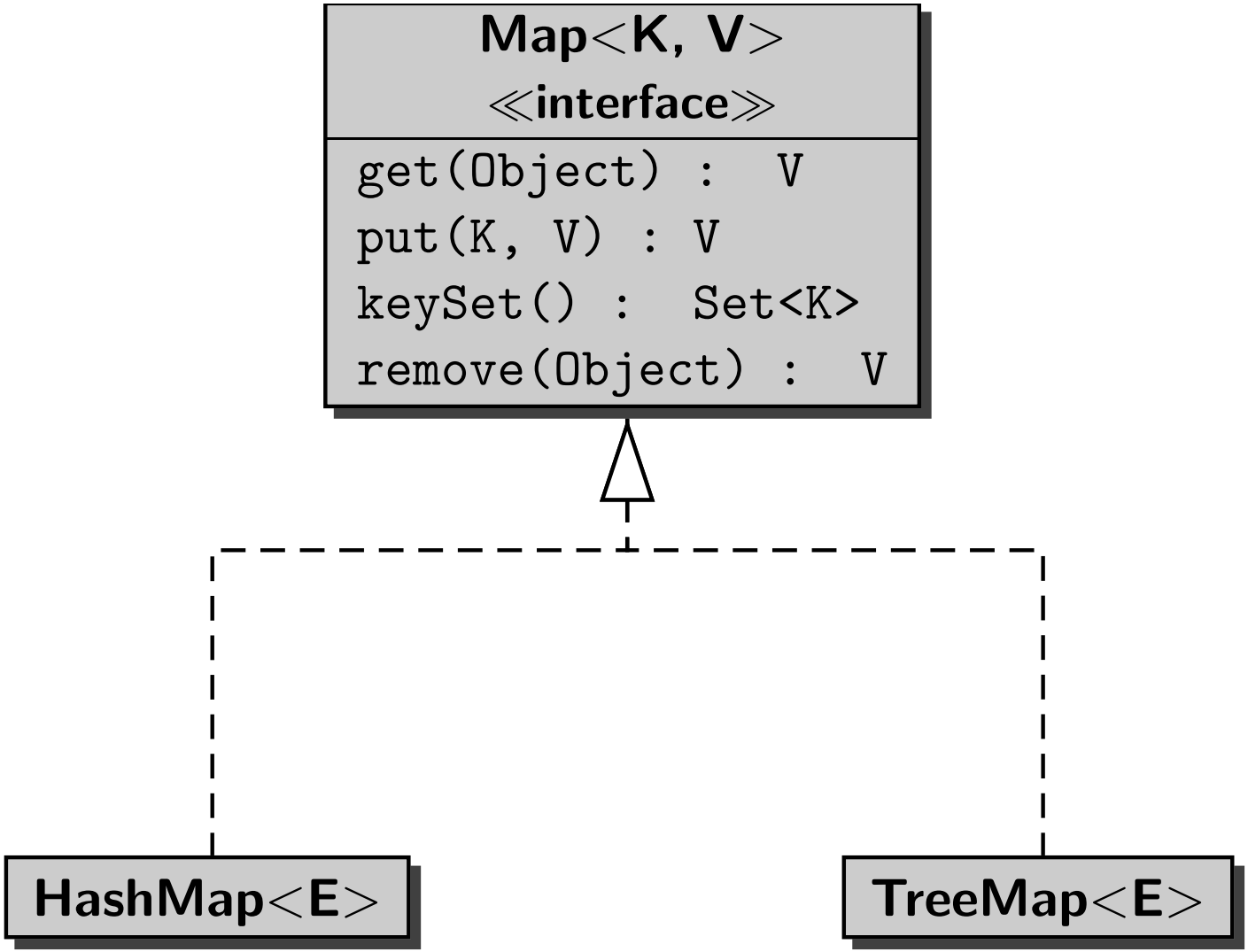
Problem

Print each song of each playlist of an iTunes library. Each song should appear on a separate line. Playlists should be separated by a blank line.

Problem

Determine whether each playlist of an iTunes library contains duplicates.

Map


Map<K, V>
≪interface≫
get(Object) : V
put(K, V) : V
keySet() : Set<K>
remove(Object) : V
HashMap<E>
TreeMap<E>

Temperature app

The temperature app

- randomly selects a city in Ontario,
- reads a corresponding URL, and
- extracts the current temperature.

Temperature app

For each city, we need a corresponding URL. These can be stored in a file.

```
on-122_metric_e.html  Alexandria
on-1_metric_e.html  Algonquin Park (Brent)
on-29_metric_e.html  Algonquin Park (Lake of Two Rivers)
on-114_metric_e.html  Alliston
on-30_metric_e.html  Apsley
on-111_metric_e.html  Armstrong
on-148_metric_e.html  Atikokan
on-164_metric_e.html  Attawapiskat
...
```

Temperature app

Question

What is the most appropriate collection to store the cities and their URLs? A list, a set or a map?

Temperature app

Question

What is the most appropriate collection to store the cities and their URLs? A list, a set or a map?

Answer

A map.

Temperature app

Question

What is the most appropriate collection to store the cities and their URLs? A list, a set or a map?

Answer

A map.

Question

What are the types of the keys and values of the map?

Temperature app

Question

What is the most appropriate collection to store the cities and their URLs? A list, a set or a map?

Answer

A map.

Question

What are the types of the keys and values of the map?

Answer

`String` and `URL`.

Object Serialization

Rather than reading a string representation of an object from a file and creating the object, we can also read the object from a file directly.

```
ObjectInputStream objectInput =
    new ObjectInputStream(
        new FileInputStream("cities.dat"));
Map<String, URL> map =
    (Map) objectInput.readObject();
objectInput.close();
```

Object Serialization

Rather than writing a string representation of an object to a file, we can also save the object to a file directly.

```
ObjectOutputStream objectOutput =
  new ObjectOutputStream(
     new FileOutputStream("cities.dat"));
objectOutput.writeObject(map);
objectOutput.close();
```

Object Serialization

Question

Which objects can be serialized?

Object Serialization

Question

Which objects can be serialized?

Answer

Those objects that are an instance of a class that implements the interface `Serializable`.

Searching

Problem

Check whether a given word appears in the book entitled "The Java Language Specification, Java SE 7 Edition." The book is contained in the file `jls7.pdf`.

Question

Consider

```
List<String> words = ...;
String search = ...;
boolean found = false;
for (String word : words)
{
   found = word.equals(seach) || found;
}
```

Given that the list contains n elements, how many times is the method `equals` invoked?

Question

Consider

```
List<String> words = ...;
String search = ...;
boolean found = false;
for (String word : words)
{
   found = word.equals(seach) || found;
}
```

Given that the list contains n elements, how many times is the method `equals` invoked?

Answer

n times.

Binary search

```
int index = Collections.binarySearch(list, element);
```

- Precondition: the list is sorted.
- If the element is contained in the list then the method returns the index at which the element can be found.
- If the element is not in the list then the method returns -1.

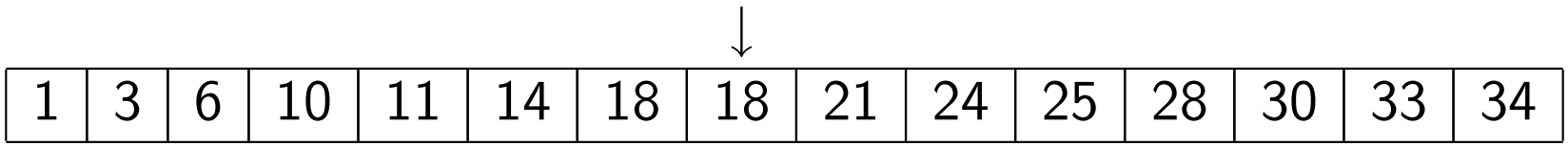
Binary search

```
final int ELEMENT = 11;
int index = Collections.binarySearch(list, ELEMENT);
```

1	3	6	10	11	14	18	18	21	24	25	28	30	33	34

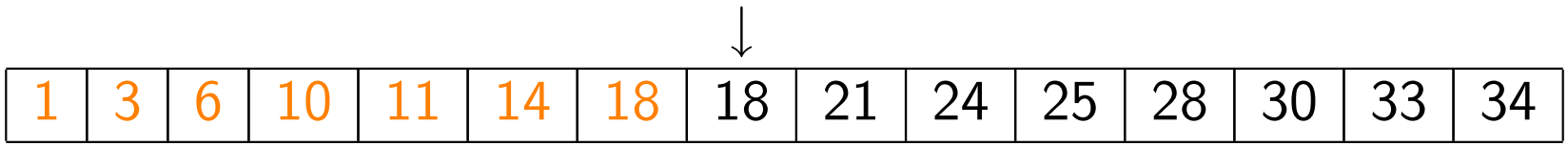
Binary search

```
final int ELEMENT = 11;
int index = Collections.binarySearch(list, ELEMENT);
```



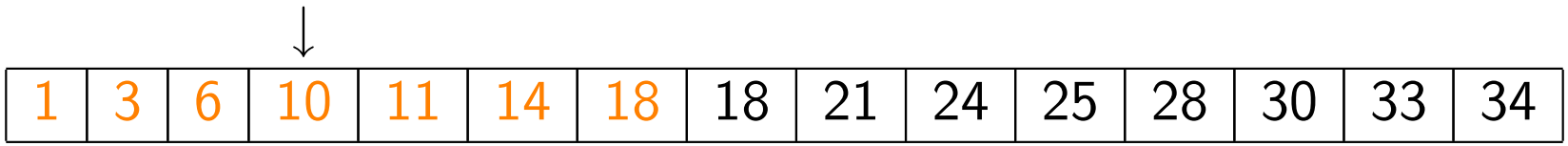
Binary search

```
final int ELEMENT = 11;
int index = Collections.binarySearch(list, ELEMENT);
```



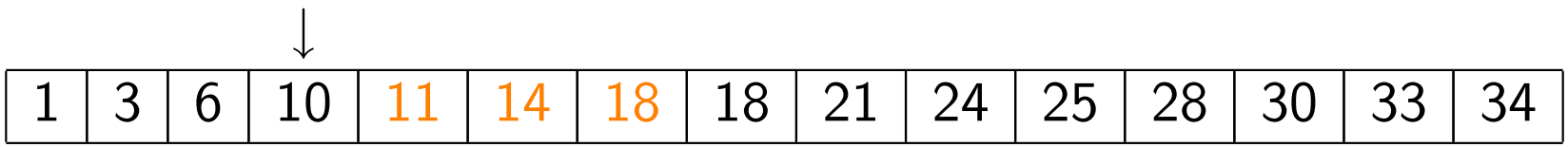
Binary search

```
final int ELEMENT = 11;
int index = Collections.binarySearch(list, ELEMENT);
```



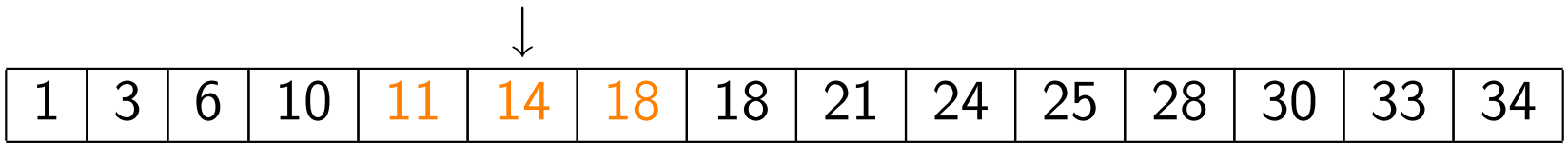
Binary search

```
final int ELEMENT = 11;
int index = Collections.binarySearch(list, ELEMENT);
```



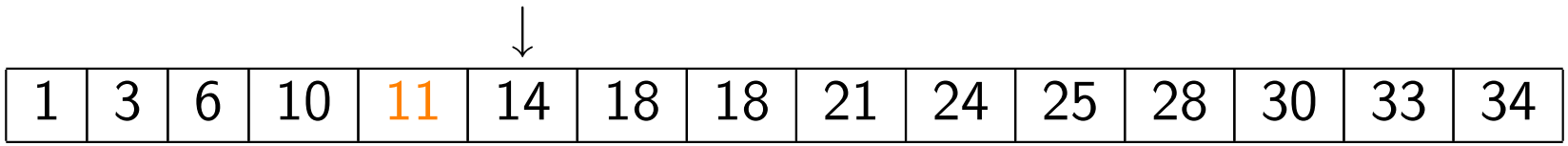
Binary search

```
final int ELEMENT = 11;
int index = Collections.binarySearch(list, ELEMENT);
```



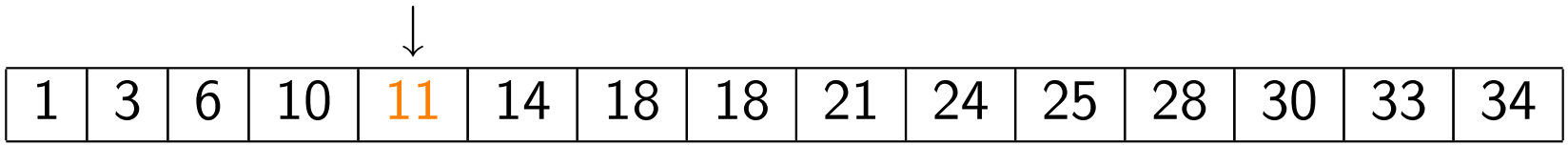
Binary search

```
final int ELEMENT = 11;
int index = Collections.binarySearch(list, ELEMENT);
```



Binary search

```
final int ELEMENT = 11;
int index = Collections.binarySearch(list, ELEMENT);
```



`index` gets assigned the value 4.

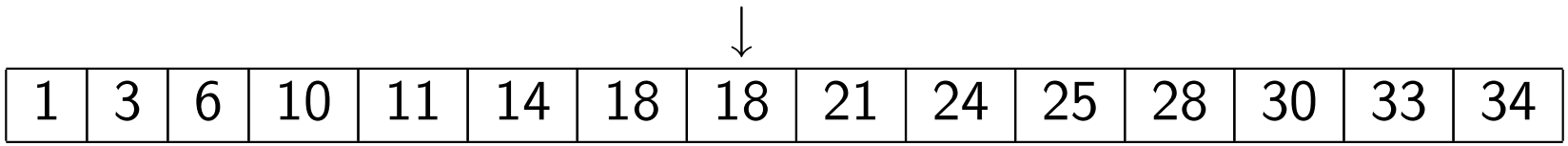
Binary search

```
final int ELEMENT = 32;
int index = Collections.binarySearch(list, ELEMENT);
```

1	3	6	10	11	14	18	18	21	24	25	28	30	33	34

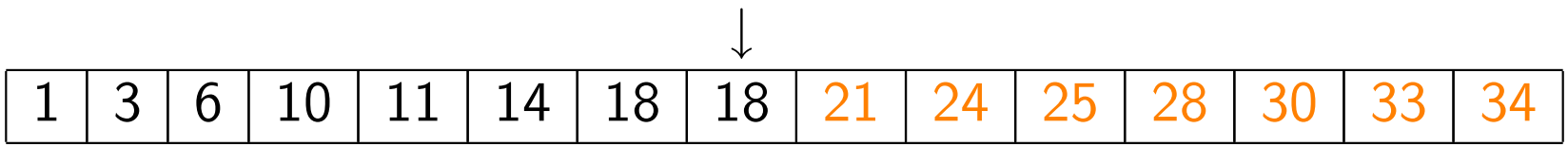
Binary search

```
final int ELEMENT = 32;
int index = Collections.binarySearch(list, ELEMENT);
```



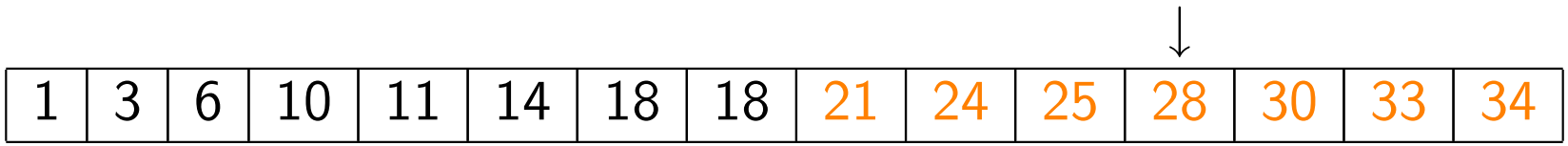
Binary search

```
final int ELEMENT = 32;
int index = Collections.binarySearch(list, ELEMENT);
```



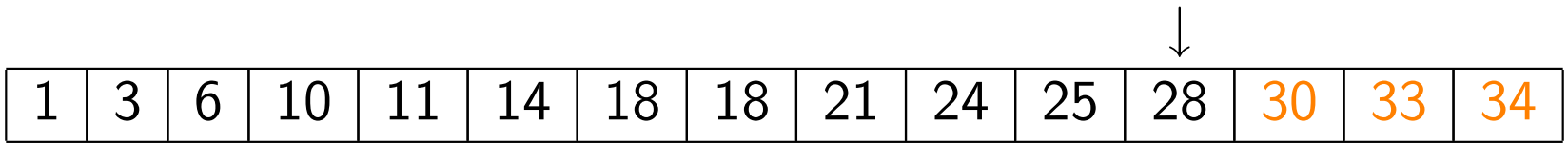
Binary search

```
final int ELEMENT = 32;
int index = Collections.binarySearch(list, ELEMENT);
```



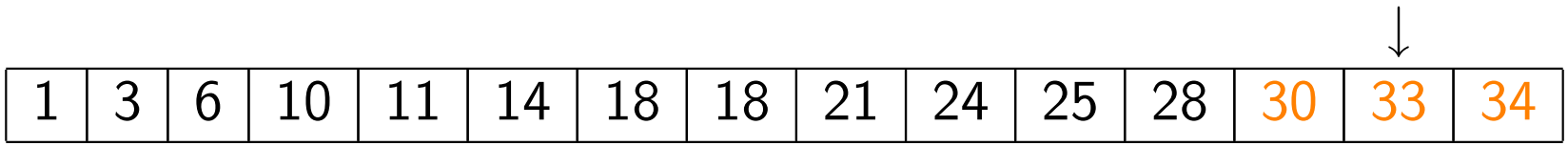
Binary search

```
final int ELEMENT = 32;
int index = Collections.binarySearch(list, ELEMENT);
```



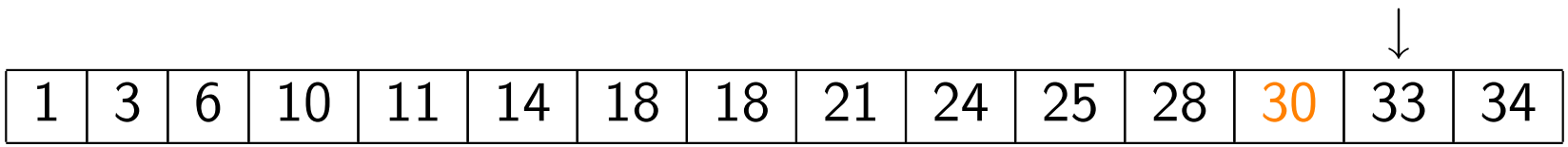
Binary search

```
final int ELEMENT = 32;
int index = Collections.binarySearch(list, ELEMENT);
```



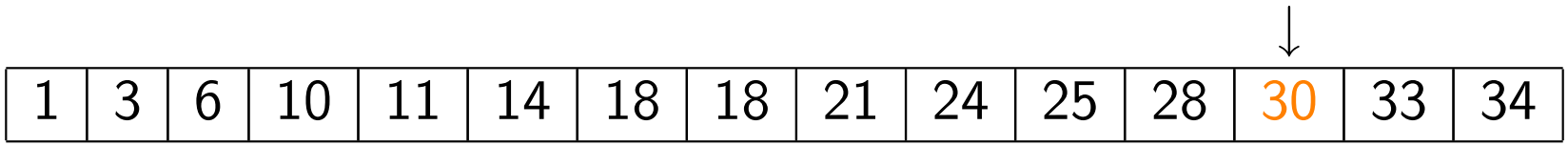
Binary search

```
final int ELEMENT = 32;
int index = Collections.binarySearch(list, ELEMENT);
```



Binary search

```
final int ELEMENT = 32;
int index = Collections.binarySearch(list, ELEMENT);
```



`index` gets assigned the value -1.

System.nanoTime()

This method returns the current value of the running Java Virtual Machine's high-resolution time source, in nanoseconds.

```
long start = System.nanoTime();
...
long stop = System.nanoTime();
// stop - start is an estimate of the number of
// nanoseconds it took to execute ...
```

Binary search

Question

Consider

```
int index = Collections.binarySearch(list, ELEMENT);
```

Given that the list contains n elements, in the worst case, how many comparisons does the invocation of `binarySearch` make?

Binary search

Question

Consider

```
int index = Collections.binarySearch(list, ELEMENT);
```

Given that the list contains n elements, in the worst case, how many comparisons does the invocation of `binarySearch` make?

Answer

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c								

Binary search

Question

Consider

```
int index = Collections.binarySearch(list, ELEMENT);
```

Given that the list contains n elements, in the worst case, how many comparisons does the invocation of `binarySearch` make?

Answer

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1							

Binary search

Question

Consider

```
int index = Collections.binarySearch(list, ELEMENT);
```

Given that the list contains n elements, in the worst case, how many comparisons does the invocation of `binarySearch` make?

Answer

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2						

Binary search

Question

Consider

```
int index = Collections.binarySearch(list, ELEMENT);
```

Given that the list contains n elements, in the worst case, how many comparisons does the invocation of `binarySearch` make?

Answer

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2	2					

Binary search

Question

Consider

```
int index = Collections.binarySearch(list, ELEMENT);
```

Given that the list contains n elements, in the worst case, how many comparisons does the invocation of `binarySearch` make?

Answer

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2	2	3				

Binary search

Question

Consider

```
int index = Collections.binarySearch(list, ELEMENT);
```

Given that the list contains n elements, in the worst case, how many comparisons does the invocation of `binarySearch` make?

Answer

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2	2	3	3	3	3	4

Answer continued

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2	2	3	3	3	3	4

Answer continued

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2	2	3	3	3	3	4
2^{c-1}								

Answer continued

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2	2	3	3	3	3	4
2^{c-1}	1	2	2	4	4	4	4	8

Answer continued

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2	2	3	3	3	3	4
2^{c-1}	1	2	2	4	4	4	4	8

$2^{c-1} \leq n$

Answer continued

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2	2	3	3	3	3	4
2^{c-1}	1	2	2	4	4	4	4	8

$2^{c-1} \leq n$

$\log_2(2^{c-1}) \leq \log_2(n)$

Answer continued

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2	2	3	3	3	3	4
2^{c-1}	1	2	2	4	4	4	4	8

$2^{c-1} \leq n$
$\log_2(2^{c-1}) \leq \log_2(n)$
$c - 1 \leq \log_2(n)$

Answer continued

Let n be the number of elements of the list and c the number of comparisons needed in the worst case.

n	1	2	3	4	5	6	7	8
c	1	2	2	3	3	3	3	4
2^{c-1}	1	2	2	4	4	4	4	8

$2^{c-1} \leq n$
$\log_2(2^{c-1}) \leq \log_2(n)$
$c - 1 \leq \log_2(n)$
$c \leq \log_2(n) + 1$

Sorting

Fact

Sorting a list of n elements needs approximately $n \log(n)$ comparisons.