Check08C

Number of students enrolled in the course: 173

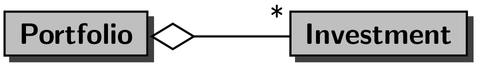
Number of students that eChecked Check08C: 24 (14%)

Check09C

Number of students enrolled in the course: 173

Number of students that eChecked Check08C: 21 (12%)

Collection



A `Portfolio` is a collection of `Investments`.

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.

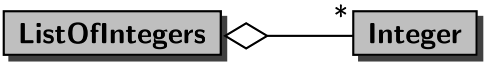
Question

Are the elements of a list ordered?

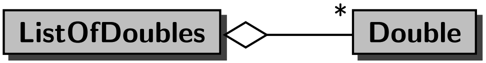
Answer

Yes.

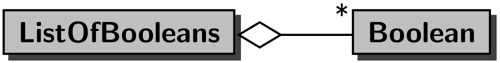
Number of eChecks Submitted each Week



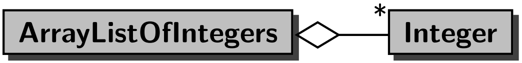
Score for each Test



Attendance

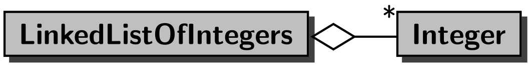


Number of eChecks Submitted each Week



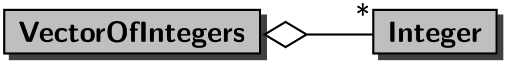
The list is implemented by means of an array.

Number of eChecks Submitted each Week



The list is implemented by means of a “links.”

Number of eChecks Submitted each Week



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

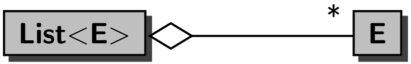
Lists

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

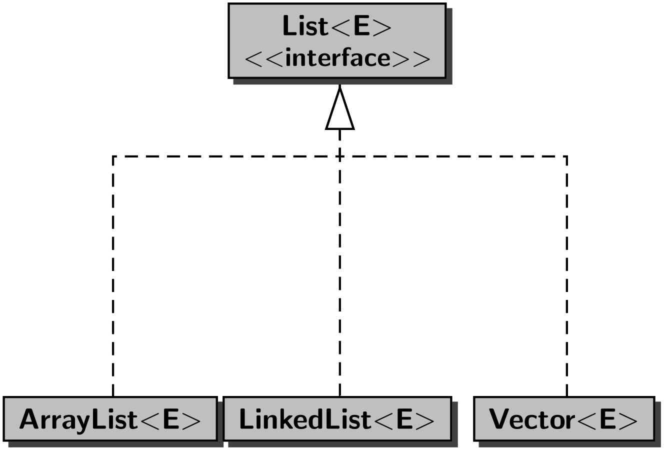
To abstract from the type of the elements of the list, we exploit generics.



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 eChecks Submitted each Week

```
final int ECHECKS = 11;
List<Integer> submissions =
   new ArrayList<Integer>(ECHECKS);
```

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

Number of eChecks Submitted each Week

```
final int ECHECKS = 11;
List<Integer> submissions =
   new ArrayList<Integer>(ECHECKS);
```

- 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 eChecks Submitted each Week

```
final int ECHECKS = 11;
List<Integer> submissions =
   new ArrayList<Integer>(ECHECKS);
```

- 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

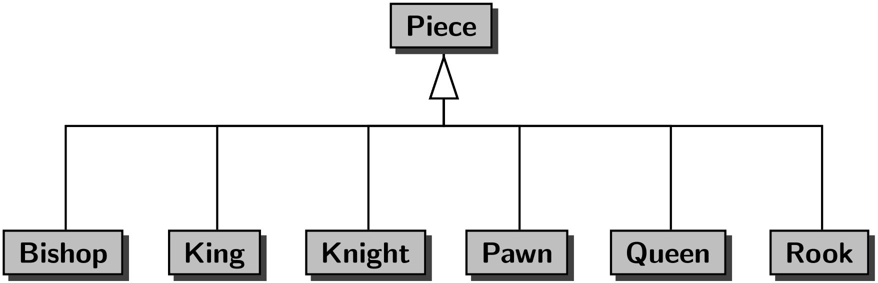
n	`ArrayList`	`LinkedList`	`Vector`
10^5	9	12	14
10^6	47	92	113
10^7	442	824	1041
2×10^7	913	1,650	2,076
3×10^7	1,350	143,616	3,230
4×10^7	2,527		4,103
5×10^7	2,689		6,119

ArrayList, LinkedList or Vector?

- Adding to or deleting from the beginning of a list takes LinkedList $O(1)$, whereas it takes ArrayList and Vector $O(n)$ where n is the size of the list.
- Adding and deleting while traversing a list takes LinkedList $O(1)$, whereas it takes ArrayList and Vector $O(n)$ where n is the size of the list.
- In most other cases, ArrayList outperforms LinkedList and Vector.

More about this in Fundamentals of Data Structures (CSE2011).

Chess Pieces

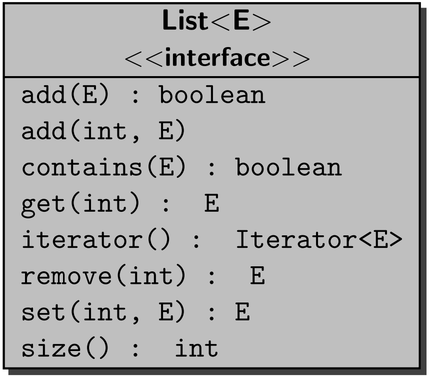


Row of a Chess Board

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

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

Methods of List



List<E>
<<**interface**>>

```
add(E) : boolean
add(int, E)
contains(E) : boolean
get(int) :  E
iterator() :  Iterator<E>
remove(int) :  E
set(int, E) : E
size() :  int
```

Row of a Chess Board

Question

Create an empty row of a chess board.

Row of a Chess Board

Answer

```
final int ROWS = 8;
List<Piece> row = new ArrayList<Piece>(ROWS);
for (int r = 0; r < ROWS; r++)
{
   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

```
row.set(0, new Rook(Color.BLACK));
row.set(ROWS - 1, new Rook(Color.BLACK));
```

Row of a Chess Board

Question

Place a white pawn on each square of the row.

Row of a Chess Board

Answer

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

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().trim());
```

Chess Board

Question

How do you represent a chess board?

Chess Board

Answer

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

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

Chess Board

Question

Create an empty chess board.

Chess Board

Answer

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

Chess Board

Question

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

Chess Board

Answer

```
List<Piece> row = board.get(0);
row.set(0, new Rook(Color.BLACK));
row.set(ROWS - 1, new Rook(Color.BLACK));
```

Chess Board

Question

Place a white pawn on each square of the one but last row of the board.

Chess Board

Answer

```
for (int r = 0; r < ROWS; r++)
{
   List<Piece> row = board.get(COLUMNS - 2);
   row.set(r, new Pawn(Color.WHITE));
}
```

Chess Board

Question

Print the board.

Chess Board

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

Java Language Specification

Question

Which word occurs most often in the “Java Language Specification”?

The ASCII version of the “Java Language Specification” is stored in the file `jls.txt`.

Print the following: `The word "..." occurs ...  times in the Java Language Specification.`

`Collections.sort`

```
public static <T extends Comparable<? super T>> void
    sort(List<T> list)
```

The method `sort` takes as argument a list of type `List<T>` where `T` satisfies

```
T extends Comparable<? super T>
```

That is, `T` implements the interface `Comparable<S>` where `S` is either `T` or any of its ancestors.

Therefore, `T` contains the method `compareTo(S)` and, hence, can compare elements of type `T` (since `T` is-an `S`).

This method takes $O(n \log(n))$, where n is the size of the list.

Java Language Specification

Question

Prompt the user for a word using the prompt `Enter word: ` and determine if that word occurs in the “Java Language Specification”?

Print the following: `The word "..." does not occur/occurs in the Java Language Specification.`

Collections.binarySearch

```
public static <T> int binarySearch(List<? extends
    Comparable<? super T>> list, T element)
```

The method `binarySearch` takes as argument a list of type

```
List<? extends Comparable<? super T>>
```

and an element of type `T` and returns the index of the element, if it is contained in the list.

This method takes $O(\log(n))$, where n is the size of the list.

Sets

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?

Sets

Question

May a set contain duplicates?

Answer

No.

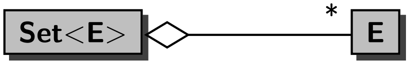
Question

Are the elements of a set ordered?

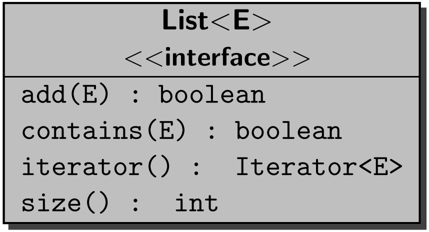
Answer

No.

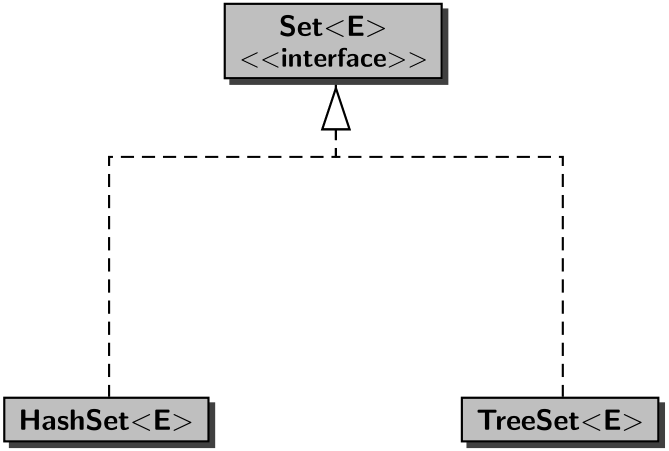
Sets



Methods of Set



Sets



Java Language Specification

Question

Prompt the user for a word using the prompt `Enter word:` and determine if that word occurs in the “Java Language Specification”?

Print the following: `The word "..." does not occur/occurs in the Java Language Specification.`

Remove the duplicates from the collection.