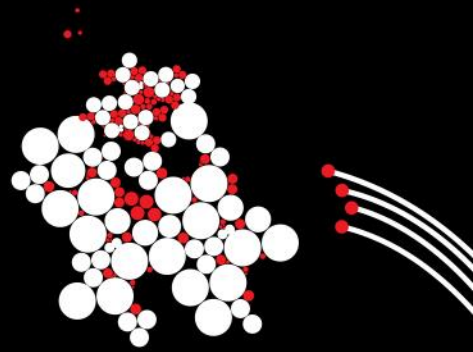
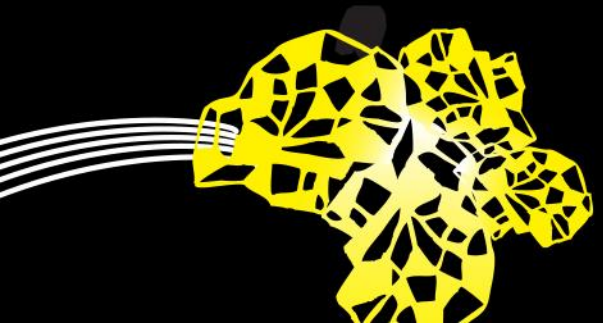
UNIVERSITY OF TWENTE.



The Generic List Interface

Topic of Software Systems (TCS module 2)

Lecturer: Marieke Huisman



List<E> QUERIES

PACKAGE java.utils

- Number of elements: `public int size()`
  - Postcondition: `ensures \result >= 0;`
- Is list empty? `public boolean isEmpty()`
  - Postcondition: `ensures \result == (this.size() == 0);`
- Get element at index i: `public E get(int i)`
  - Precondition: `requires 0 <= i && i < this.size();`

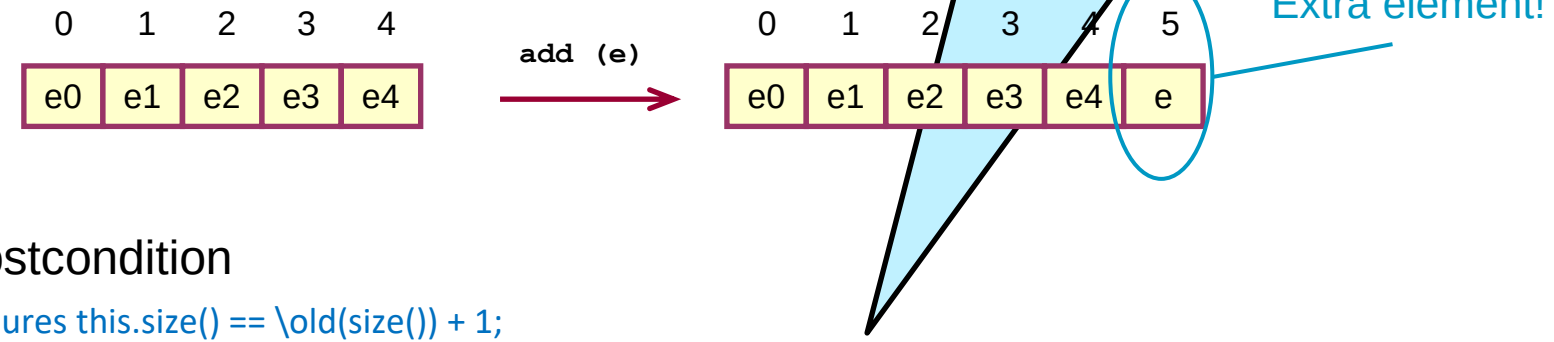
List size cannot be negative

Index *i* should be inside the list!

List<E> COMMANDS

List
- grows one position,
- new element *e* is added to the end of the list
- all other elements remain the same

Append element: public void add(E e)



Postcondition

```
ensures this.size() == \old(size()) + 1;
ensures this.get(this.size() - 1).equals(e);
ensures (\forall int i; 0 <= i && i < \old(size());
            this.get(i).equals(\old(get(i)));
```

List<E> COMMANDS

Set element at index i: `public void set(int i, E e)`

Precondition:

```
requires 0 <= i && i < this.size();
```

Postcondition:

```
ensures this.size() == old.size();
ensures this.get(i).equals(e);
ensures (\forall int j; 0 <= j & j < this.size() & j != i;
                        this.get(j).equals(\old(get(j))));
```

List size does not change, new element *s* is placed in position *i* and all other elements remain the same

OTHER METHODS

- Index of first occurrence of element: `public int indexOf(E e)`
  - returns -1 if this list does not contain the element
- Does the list contain an element? `public boolean contains(E e)`
- Removes first occurrence of element: `public boolean remove(E e)`

And many more (see Java 11 API)

List<E> USAGE

- Interfaces cannot be used to instantiate objects, so we need classes
- Select a list implementation that fits your application!

```
List<Student> sl1 = new ArrayList<Student>(); // ok in most cases!
List<Student> sl2 = new LinkedList<Student>();
```

- By defining your reference as interface List<E> you can change the implementation later if necessary
- And don't forget to import the classes java.utils.List, java.utils.ArrayList, etc.

List<E> USAGE

FOR COMPLETENESS

```
List<Integer> values = new ArrayList<Integer>();
List<Room> rooms = new ArrayList<Room>();
```

- Read values from a list

```
int i = values.get(3); // copy 4th value of values to i
Room r = rooms.get(2); // copy 3th reference of rooms to r
```

- Modify values from a list

```
values.set(3, i);  // copy i to 4th position of values
rooms.set(2, r);   // copy reference r to 3th position of rooms
```

ITERATION OVER LISTS

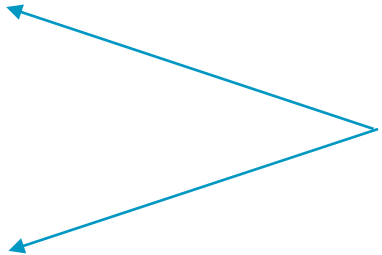
SHORTCUTS

- Java offers special syntax (shortcuts) to facilitate iteration over lists

```
java.util.List<Student> list;
for (Student s : list) {
  System.out.println(s);
}
```

- Alternative: Iterator Class

```
Iterator<Student> it = list.iterator();
while (it.hasNext()) {
  Student s = it.next();
  System.out.println(s);
}
```



Warning: Avoid changing list size while iterating over it!
For example, avoid calling list.add(s);