

A pointillist painting depicting a park scene. In the foreground, several trees with blossoming branches in shades of yellow, red, and white are scattered across a grassy field. A small, simple bridge with a single arch spans a stream or path in the middle ground. In the background, there's a building with a prominent red roof and a tall, thin chimney emitting smoke. The sky is filled with a dense, textured pattern of various colors.

# A01 Abstract Data Types: Lists

ADTs

List as an ADT  
A List interface

# Abstract Data Types (ADTs)

Abstract data types\* describe the behaviour (semantics) of a data type without specifying its implementation. An ADT is thus **abstract**, not concrete.

- A **container** is a very general ADT, a holder of objects.
- A **list** is an example of a more specific container ADT.

\* Not to be confused with: *Algebraic Data Type*.

# The List ADT

The **list** ADT is a container known mathematically as a finite sequence of elements. A list has these fundamental properties:

- duplicates are allowed
- order is preserved

A list may\* support operations such as these:

- *create*: construct an empty list
- *add*: add an element to the list
- *is empty*: test whether the list is empty

\* The operations a given ADT must support will vary depending on the author / library

# Our List Interface

We will explore lists using a simple interface:

```
public interface List<T> {  
    void add(T value);  
    T get(int index);  
    int size();  
    T remove(int index);  
    void reverse();  
}
```

```
void add(T value);
```



```
T get(int index);
```



```
int size();
```



```
T remove(int index);
```



```
void reverse();
```



```
String toString();
```

