

Abstract Data Types (ADTs)

Lecturer: Arran Stewart

Outline

- ▶ What is an abstract data type?
- ▶ What do we use them for?

Abstraction

- ▶ Abstract Data Types allow us to *abstract away* from implementation detail.
- ▶ We will discuss some examples of this.

Operations

If we define a data structure in terms of what we can *do* with it – rather than in terms of exactly how it is arranged in memory – then we have specified the *operations* we can perform on it.

Such a specification is an Abstract Data Type.

Object-oriented languages

- ▶ Object-oriented programming is based around the concept of abstract data types.
- ▶ Java is an object-oriented programming language and Java classes are ideal for implementing ADTs.

What is needed to make an ADT?

- ▶ Variables for holding data
 - ▶ usually hidden from the user
- ▶ Operations that can be performed on the data
 - ▶ these must be available to the user

Examples

- ▶ We will consider examples of these requirements for ADTs.

Information hiding

- ▶ ADTs support *information hiding*.
- ▶ Variables can be made private, meaning there is no access to the variables by users of a class.
- ▶ Methods can be made public, so users of the class can create and manipulate the data structure.

Why information hiding?

- ▶ Information hiding is good programming practice –
- ▶ We can change how data is stored and the way the methods are implemented without changing the external functionality.

Modularity

- ▶ ADTs provide *modularity*. What does this mean?
- ▶ It means that – as long as the public parts of the ADT have been clearly specified – other parts of our program can be developed, maintained or upgraded independently of our ADT.
- ▶ They don't depend on *internal details* of the ADT, so we can even start writing them before the ADT is done.
- ▶ Or, if we change our mind about how to implement the ADT – as long as the public interface remains the same, this won't affect other parts of our software.

Other benefits

In object-oriented languages such as Java, ADTs provide other benefits such as:

- ▶ *polymorphism* (the same operation can be applied to different types)
- ▶ *inheritance* (where subclasses adopt from parent classes)

We will see these features in more detail later in the course.