

## Specifying an ADT

---

---

---

---

---

---

---

## Abstraction (revisited)

- A tool to manage complexity; highlight the interface to a “thing” while hiding its implementation details
- Applied to data types  $\Rightarrow$  Abstract Data Type
  - The *specification view* of a type: no implementation details, just values and behavior

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-2

---

---

---

---

---

---

---

## Support

`class` SortedArrayList **implements** SortedListInterface

Java allows us to define classes and interfaces.

- Interfaces form a contract between the class and the outside world, and this contract is enforced at build time by the compiler. If your class claims to implement an interface, all methods defined by that interface must appear in its source code before the class will successfully compile.

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-3

---

---

---

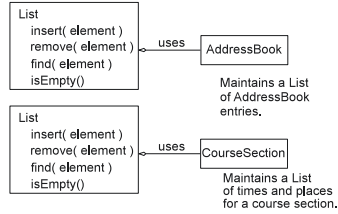
---

---

---

---

## Abstraction: Collection Example



A List has many uses. A client (user) accesses a List through its public interface (API) without any knowledge of how the List is implemented.

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-4

---

---

---

---

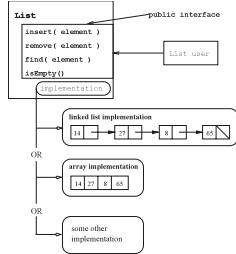
---

---

---

---

## Abstraction: Collection Example



What does a user of a List need to know about it? Its behavior as described in its API and related documentation.

Possible implementations of the List ADT. The implementation details are hidden from the user of the List, who relies on the List's public interface.

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-5

---

---

---

---

---

---

---

---

## Abstraction by Specification

- We need to describe the nature of a computation without relying on reading the body of the method that performs the computation.
  - Makes it easier to choose which computation to use.
  - Makes it easier to distribute the development tasks.
  - Separates design from implementation

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-6

---

---

---

---

---

---

---

---

## Strategy

- We can accomplish abstraction by specification if we associate with a method sufficient information to allow others to understand and use that method without looking at the body.
- Design/Programming by contract  
{Pre}C{Post}

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-7

---

---

---

---

---

---

---

---

## Assertions

- A good way to write the information required is through assertions.
- Precondition
  - The conditions assumed to be true on entry to the operation if the operation is to execute successfully. This may include assumptions about the state of the object on entry, assumptions about the parameters passed in, and so on.
- Postcondition
  - What the operation guarantees to be true on exit and how the operation changed the state of the object. On exit from the operation, no change to the object's state may violate any of the ADT's invariants.

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-8

---

---

---

---

---

---

---

---

## Example

```
float sqrt (float value) {  
    //precondition : value>=0  
    // postcondition : returns an approximation for the square  
    //root of value, ans>=0 && ans*ans<=value  
    float ans=value/2.0f;  
    int i=1;  
    while (i<7) {  
        ans = ans-((ans*ans-value)/(2.0f*ans));  
        i++;  
    }  
    return ans;  
}
```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-9

---

---

---

---

---

---

---

---

## Rules

- After the execution of the method, we can assume the postcondition holds provided the precondition held when the call was made.
- Typically in design by contract it is the users' job to insure the contract (preconditions) is met. However, pragmatics suggests that any operation which modifies the state of the object take the responsibility of checking the precondition and throw an exception as needed.
- We can only assume those properties that can be inferred from the postcondition.

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-10

---

---

---

---

---

---

---

---

## Discussion

- Precondition states the constraints on use. Usually needed which the method does not work for all inputs (partial).
- The postcondition clause describes the behavior of the method. The clause must define what results are produced and what modifications are to the inputs. Consider Side-effects

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-11

---

---

---

---

---

---

---

---

## Specifying an ADT

ADT name  
ADT description A brief description of the type.  
ADT invariants Characteristics that must always be true of an instance of this type.  
ADT attributes Represent the state of an instance of the ADT.  
ADT operations Define the behavior of the ADT and the interface available to clients.

Include the following for each operation:

operation: The purpose of the operation – its intended semantics.  
pre-conditions: The conditions assumed to be true on entry to the operation if the operation is to execute successfully. This may include assumptions about the state of the object on entry, assumptions about the parameters passed in, and so on.  
post-conditions: What the operation guarantees to be true on exit and how the operation changed the state of the object. On exit from the operation, no change to the object's state may violate any of the ADT's invariants.  
returns: The type of value, if any, returned by the operation.  
exceptions: A description of the exceptions an operation may generate and the circumstances.

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-12

---

---

---

---

---

---

---

---

## Specification Template

```
return_type method_name( ...)  
/**  
 * Narrative Description of method  
 * precondition : States the constraints on use.  
 * postcondition : Defines the behavior.  
 * @param describe parameter used  
 * include @param for EACH parameter  
 * @return describe the return type and range of values  
 * @throws for any checked and unchecked exceptions the caller might catch  
 */
```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-13

---

---

---

---

---

---

---

---

## Example

```
public class Arrays {  
    public static int search (int [] a, int x) {  
        // precondition: if x is in a, returns index where x is  
        // stored, otherwise return -1  
  
        public searchSorted (int []a, int x) {  
            // precondition : a is sorted in ascending order.  
            // postcondition : If x is in a, returns index where x is  
            // stored, otherwise return -1  
        }  
    }  
}
```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-14

---

---

---

---

---

---

---

---

## Specification and Exceptions

```
public static int search (int [] a, int x) throws  
    NullPointerException, NotFoundException  
// precondition : a is sorted in ascending order  
// postcondition : if a == null throws  
    NullPointerException; else if x is not in a  
    throws NotFoundException; else returns i such  
    that x==a[i].
```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-15

---

---

---

---

---

---

---

---

```

public static int min (int [] a) throws
    NullPointerException, EmptyException {
    // postcondition : if a == null throws
    NullPointerException else if a is empty throws
    EmptyException else returns the minimum
    value of a.

```

```

int m;
try {m=a[0];}
catch(IndexOutOfBoundsException e) {
    throw new EmptyException(" Arrays.min");}
for (int I=1;i<a.length; i++) if (a[i]<m) m=a[i];
return m;
}

```

---

---

---

---

---

---

---

---

## Partial Specification of the Rectangle ADT

### Description

A rectangle is a four-sided shape in which opposite sides are parallel and equal in size. The length of a side must be greater than 0. A rectangle has four right angles.

### Invariants

1. Opposite sides are of equal size.
2. Length and height must be greater than 0.

### Attributes

*DEFAULT\_SIZE* a constant, the default size for the dimensions  
*length* size of the "top" and "bottom" sides  
*height* size of the "left" and "right" sides  
*surface area* the surface area of the rectangle  
*perimeter* the perimeter of this rectangle

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-17

---

---

---

---

---

---

---

---

## Partial Specification of the Rectangle ADT

### Operations

constructor()

pre-condition: none

responsibilities: default constructor—creates a rectangle with *length* and *height* set to *DEFAULT\_SIZE*

post-condition: the rectangle is initialized to the default values

returns: nothing

constructor( length, height )

pre-condition: none

responsibilities: creates a rectangle with *length* length and *height* height; if a dimension is invalid, the default size is used

post-condition: the rectangle is initialized to client-supplied values, if valid, or the default values otherwise

returns: nothing

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-18

---

---

---

---

---

---

---

---

## Partial Specification of the Rectangle ADT

getLength()  
pre-condition: none  
responsibilities returns this rectangle's *length*  
post-condition: the rectangle is unchanged  
returns: the *length* of the rectangle

setLength( newLength )  
pre-condition: none  
responsibilities resets *length* to newLength if newLength is valid; otherwise does nothing  
post-condition: rectangle's *length* field is updated if newLength is valid  
returns: nothing

getSurfaceArea()  
pre-condition: none  
operation: compute the surface area of this rectangle  
post-condition: the rectangle is unchanged  
returns: the surface area of the rectangle

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

1-19

---

---

---

---

---

---

---

---