Abstract Data Types

Week 2

Stephen Naicken

28th May 2018

African Leadership University

Updates

Assessment

- 50% exam
- 50% coursework
  - 60% from a programming assignment (W1-5).
  - 20% from a problem set (W6-7).
  - 20% online quizzes on GCU Learn.
- Assignment release and submission dates TBC.

Lab Groups

Please visit `http://bit.ly/2JaAMNT` to find out which lab session you will be required to attend. **Attendance is compulsory**.

Week 1: Recap

Bag

In week 1, we implemented a model of a bag of Lego bricks in Java. Students did the following:

- Initial implementation using a single class.
- Changed the implementation to use an interface and an array.

Questions

- Why did we make use of an interface?
- Why did we switch to an array?
- How is this connected to ADTs?

Student code

Bag.java

```
public class Bag {
    List<LegoBrick> al;

    public Bag(){
        al = new ArrayList<LegoBrick>();
    }

    public void addBrick(LegoBrick newBrick){
        al.add(newBrick);
    }

    public void removeBrick(){
        if(!al.isEmpty()){
            Random rand = new Random();
            int  n = rand.nextInt(al.size());
            al.remove(n);
        } else {
            System.out.println("the␣bag␣is␣empty");
        }
    }

    public void empty(){
        al.clear();
    }
}
```

Improving the Bag class

- Our Bag can only hold one type of object, LegoBrick.
- But we might want a Bag the can hold other types of objects.
- How can we make this so?

Improving the Bag class

- Our Bag can only hold one type of object, LegoBrick.
- But we might want a Bag the can hold other types of objects.
- How can we make this so?
- We can use an interface.

Bag interface

```java
interface Bag <T>{

    // add an object of type T to the bag
    void add(T t);

    // remove an object of type T from the bag
    T remove(T t);

    // check if an object of type T is contained in the bag
    boolean contains(T t);

    // returns the number of items in the bag
    int size();

    // empty the bag
    T[] empty();

}
```

Abstract Data Type

An Abstract Data Type (ADT) defines:

- A well-specified collection of data (data state)
- A set of operations that can be performed upon the data

Abstract Data Type

An Abstract Data Type (ADT) defines:

- A well-specified collection of data (data state)
- A set of operations that can be performed upon the data

Question

Is the Bag interface an ADT?

Abstract Data Type

An Abstract Data Type (ADT) defines:

- A well-specified collection of data (data state)
- A set of operations that can be performed upon the data

Question

Is the Bag interface an ADT?

Answer

Yes, but who can explain why?

Bag ADT

A well-specified collection of data (data state)

```
    interface Bag <T>{
```

A Bag ADT that can hold objects of type T.

Bag ADT

A well-specified collection of data (data state)

```
    interface Bag <T extends Student>{
```

Question

A Bag ADT that can hold objects of type... ?.

Bag ADT

A well-specified collection of data (data state)

```
    interface Bag <? extends T>{
```

Question

A Bag ADT that can hold objects of type... ?.

Bag ADT

A set of operations that can be performed upon the data

```
// add an object of type T to the bag
void add(T t);

// remove an object of type T from the bag
T remove(T t);

// check if an object of type T is contained in the bag
boolean contains(T t);

// returns the number of items in the bag
int size();

// empty the bag
T[] empty();
```

Data Structure

- A base storage method (e.g. an array, a list, a tree, a . . .)
- One or more algorithms that are used to access or modify that data

A class that implements the ADT interface is a model of a data structure (Concrete Data Type).

BetterLegoBag

A base storage method

```java
// backing data structure to store the object
private Lego[] legos;
```

LegoBag

One or more algorithms that are used to access or modify that data

All the methods implementing the methods defined in the interface (i.e. contract).

Polymorphism and Encapsulation

Encapsulation

```
    // backing data structure to store the object
    private Lego[] legos;
```

The internals of the data structure are hidden and only exposed through the methods defined in the interface (and implemented in the class).

Polymorphism and Encapsulation

Polymorphism

```java
public class Thief {

    void steal(Bag<?> bag){
        // Thief can steal any bag!
    }
}
```

Week 2

Learning Outcomes

What you must know

- The ADT definition.
- An ADT is abstract and requires an implementation.
- The interface is the ADT.
- The implementation is a data structure.
- Know the rationale/benefits of ADTs.

Learning Outcomes

What you must be able to do

- Code a Java interface to define an ADT.
- Code an implementation of a simple ADT.

Tasks & Deliverables

GCU

1. Study week 1 and 2 of the GCULearn content.
2. Week 2 GCULearn Quiz **(Summative)**.

Tasks & Deliverables

ALU

1. Implement a Stack ADT.
2. Implement a Stack data structure.
3. Use the Stack implementation to implement one of the following:
   - 3.1 Prefix to Postfix conversion
   - 3.2 Postfix calculator
   - 3.3 Towers of Hanoi algorithm

Deadline: Tuesday 04/06/2018 @ 16hr (Hard).

Assessment

ALU

- This is individual work, no plagiarism, no collusion.
- You may be asked to perform a Kata of your code in the compulsory lab class.
- Speak natural language, not Java (or any other programming language)
- Work will assessed with respect to the learning outcomes, no grades will be given.

Feedback

ALU

- Individual feedback will be given by 18/06/2018.
- Feedback will cover code quality, algorithmic logic, efficiency and so on.
- General feedback will be given in class on 11/06/2018.