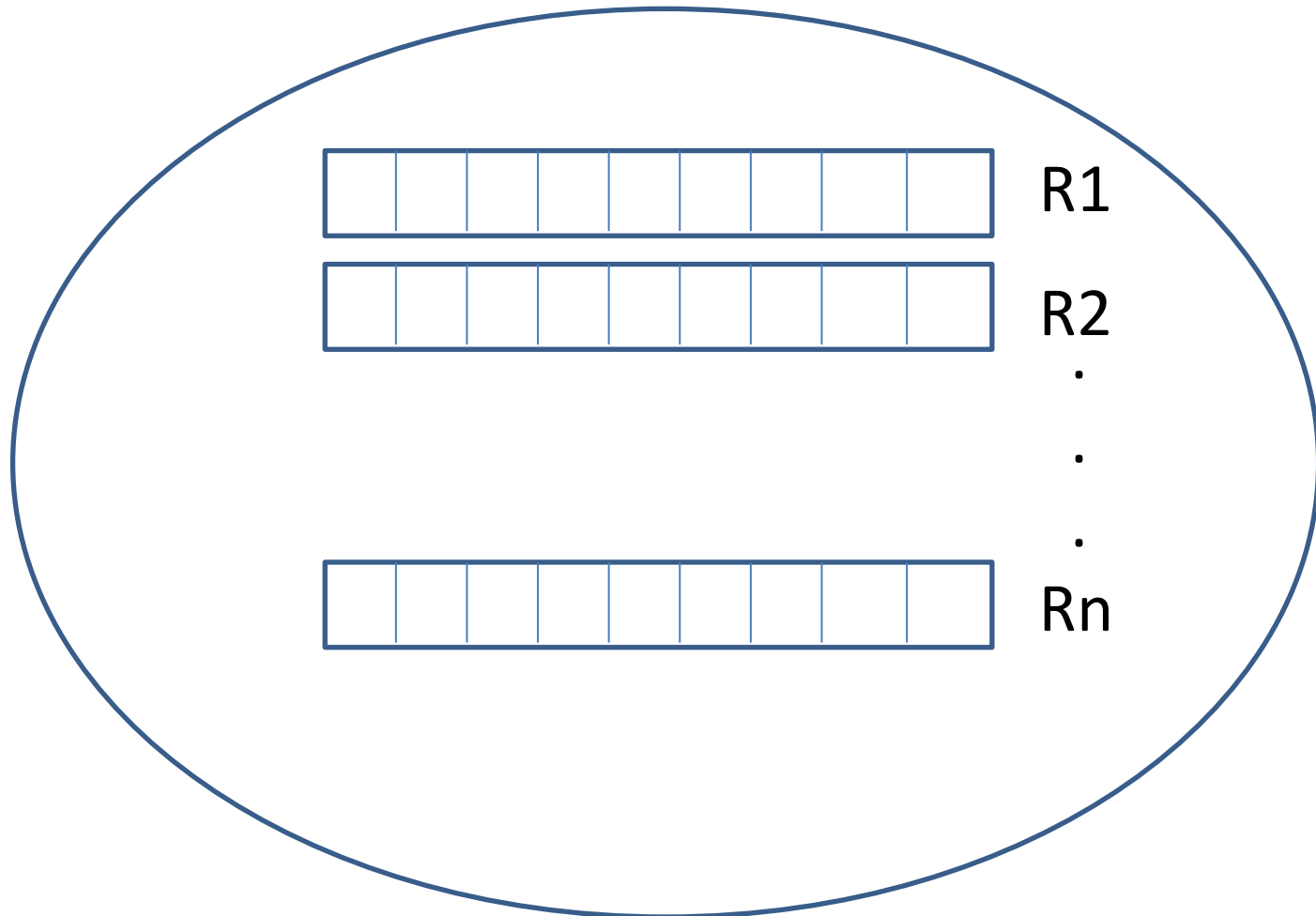


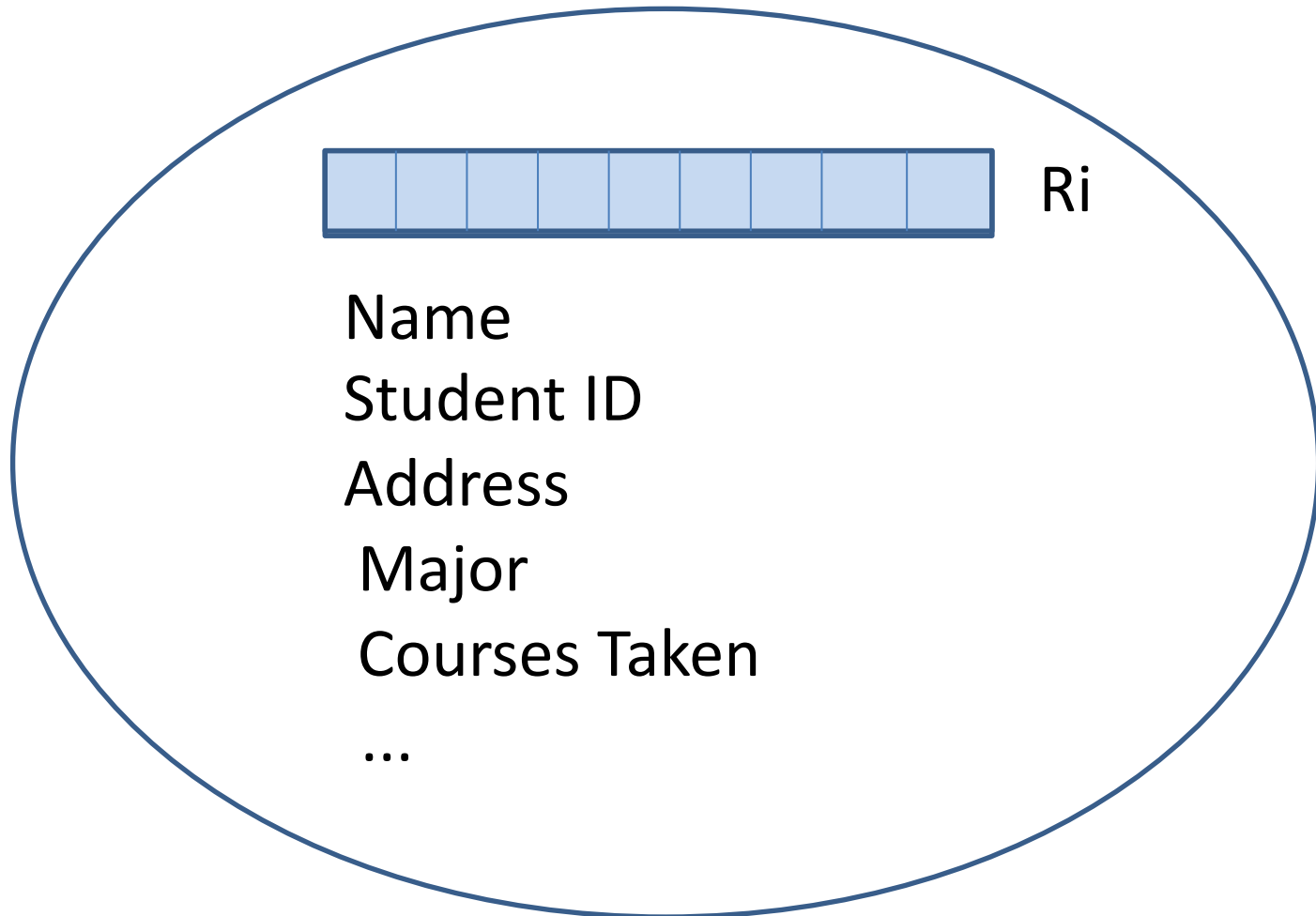
Student Information System

Data: Set of student records.



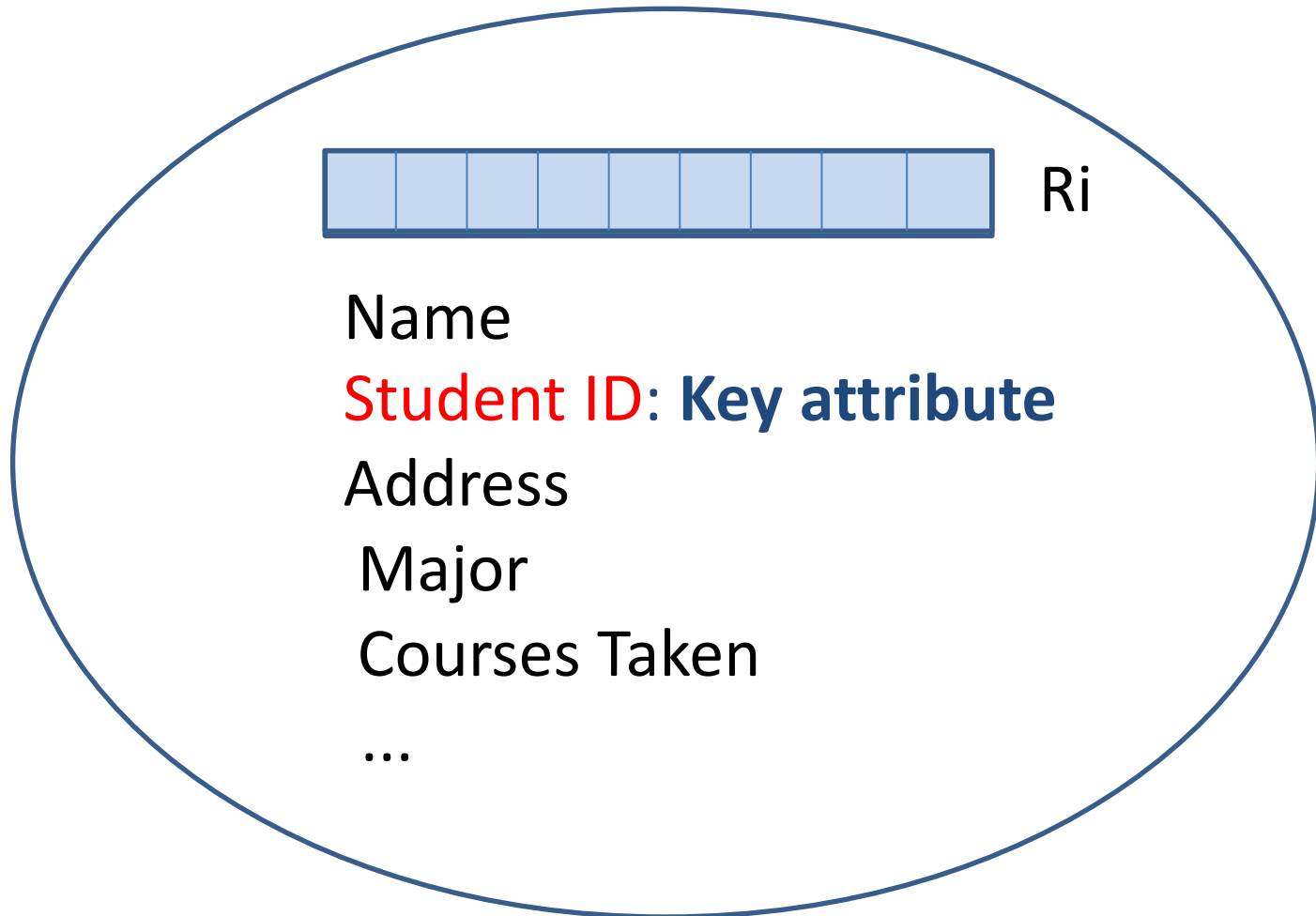
Student Information System

Each record consists of a set of attributes.



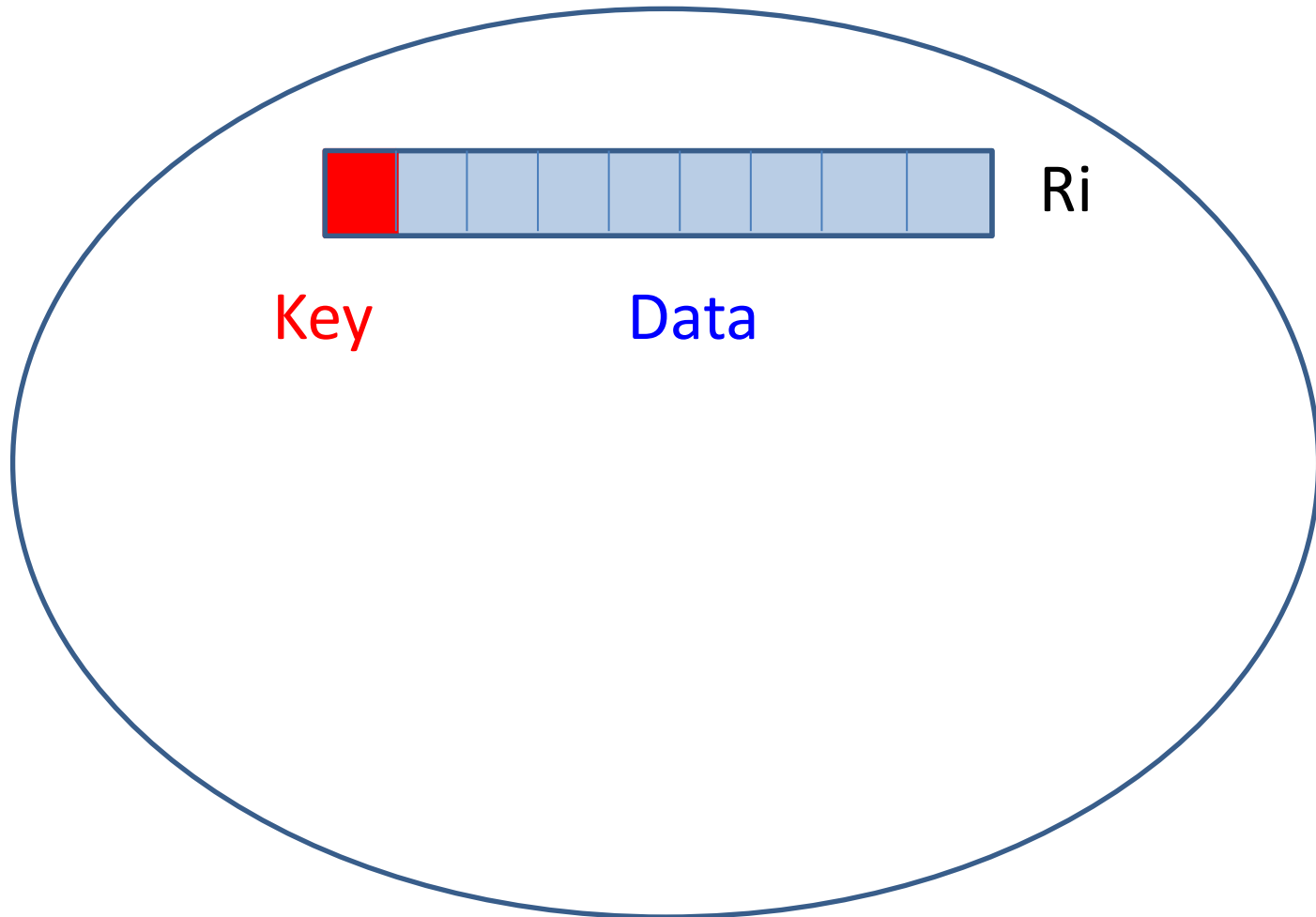
Student Information System

Each record has a unique key attribute.



Student Information System

So we can view a record as a pair (key,data)



Student Information System

Algorithm InformationSystem ()

Display user interface (UI)

While operations remain **do** {

 Read next operation from the UI

If operation is *ADD* **then**

 Read student information from UI: ID, data

 Put new record (ID,data) in the data module

else if operation is *REMOVE* **then** {

 Read ID from UI

 Remove record with key ID from data module

 }

else if operation is *TRANSCRIPT* **then** {

 Read ID from UI

 Get record from data module with given student ID

 Print transcript

 }

}

Data Structures and ADTs

Abstract Data Types (ADTs) are user defined data types.

An ADT has 2 parts:

- A name or type, specifying a set of data (e.g. Dictionary or Map).
- Descriptions of all the operations (or methods) that manipulate that type (e.g. get, put, remove)

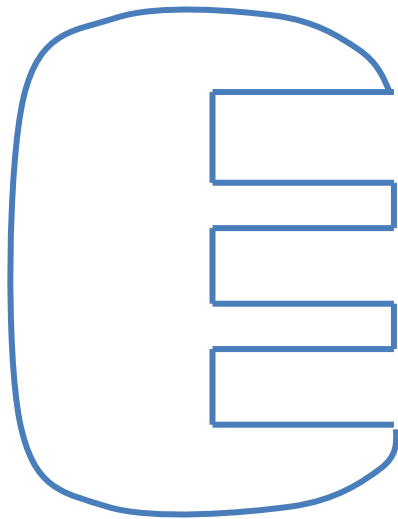
The descriptions indicate **what** the operations do, not **how** they do it.

Abstract Data Types

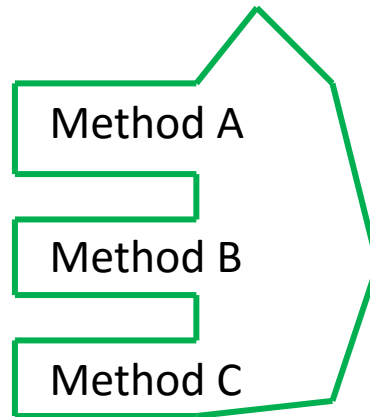
- Preferred way of designing and implementing data structures.
- Uses 2 general principles: information hiding and reusability.

Information Hiding

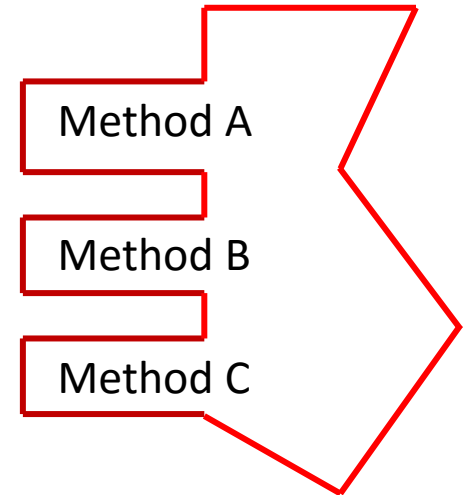
- The application that uses the data structure should not need to know details of its implementation.
- We should be able to change implementation without affecting applications that use it.
- Therefore, implementation information **should be hidden**.



Application



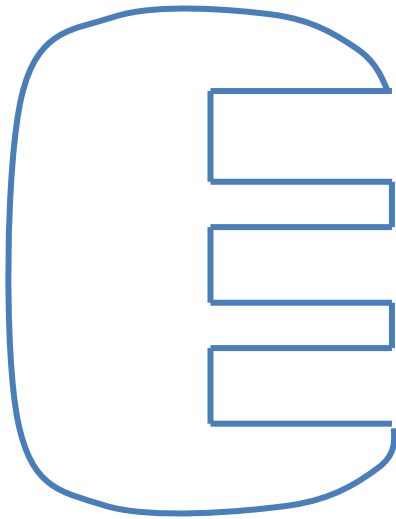
Data structure
Implementation 1



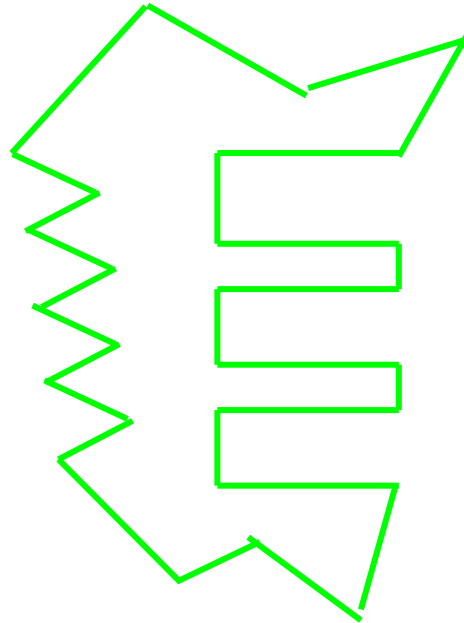
Data structure
Implementation 2

Re-usability

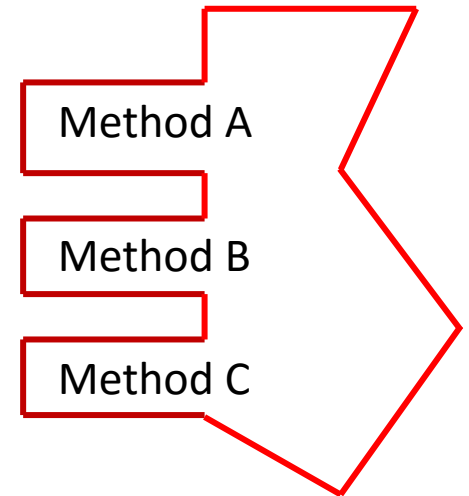
- If data structure is useful for one application, it is probably useful for others.
- Therefore, we should design it to be as re-usable as possible.



Application 1



Application 2



Data structure
Implementation

ADT Dictionary or Map

- **get (key):** returns the data associated with the given key, or null if no record has the given key
- **put(key,data):** inserts a new record with given key and data, or *ERROR* if the dictionary already contains a record with the given key
- **remove(key):** removes the record with the given key, or *ERROR* if there is no record with the given key

Java Interface for ADT Dictionary

```
public interface Dictionary <K,V>{  
    public V get(K key);  
    public void put(K key, V data) throws  
        DuplicatedKeyException;  
    public void remove(K key) throws NoKeyException;  
}
```

Java Implementation for ADT Dictionary

```
public class LinkedListDictionary <K,V> implements Dictionary {  
    private int size;  
    private DNode head, tail;  
    public LinkedListDictionary() {  
        size = 0;  
        head = new DNode(null, null,null);  
        tail = new DNode(null,null,null);  
        head.setNext(tail);  
    }  
    public V get(K key) {  
        if (size == 0) return null;  
        else { ... }  
    }  
    ...  
}
```