CO452 Programming Concepts

Lecture 5

Collections (ArrayList) and Generics

Collections

The Java collections framework is a library of classes that model data structures. Each data structure is modelled as a class that has variables and methods.

- ArrayList
- List
- LinkedList
- HashMap

Importing packages

Importing packages

Because the collection classes (ArrayList) exist in another folder (package) we have to ‘import’ the package so we can refer to the ArrayList class (create objects of it) in our class

```
import java.util.ArrayList;
/**
* Class comment…
*/
public class Student
{
      …
}
```

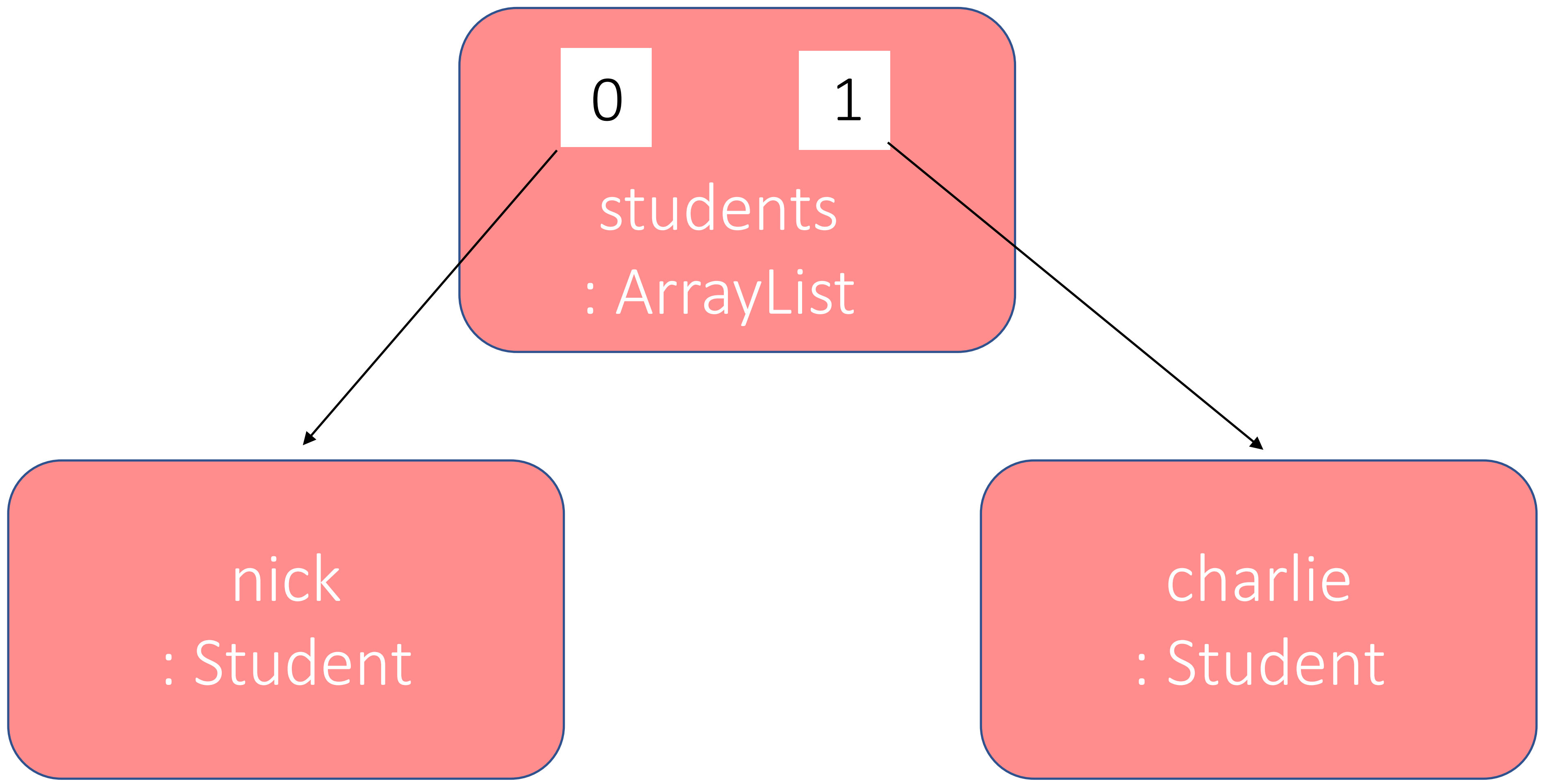
ArrayList

ArrayList

An ArrayList is a **convenient way** to store related objects in one collection.

For example, we could create a collection of student objects called ‘**students**’.

Visualisation of an ArrayList



Some methods of the ArrayList

add()

remove()

clear()

get()

size()

How to create an ArrayList

How to instantiate an object of the ArrayList

```
private ArrayList<Student> students;
private ArrayList<Product> products;
```

How to instantiate an object of the ArrayList

Scope	Class	Type of objects	object (collection)

```
private ArrayList<Student> students;
private ArrayList<Product> products;
```

How to instantiate an object of the ArrayList

Scope **Class** **Type of objects** **object (collection)**

```
private ArrayList<Student> students = new ArrayList<Student>();
private ArrayList<Product> products = new ArrayList<Product>();
```

How to instantiate an object of the ArrayList

Scope **Class** **Type of objects** **object (collection)** **Constructor**

```
private ArrayList<Student> students = new ArrayList<Student>();
private ArrayList<Product> products = new ArrayList<Product>();
```

Comparison with object syntax

Scope **Class** **Type of objects** **object (collection)** **Constructor**

```
private ArrayList<Student> students = new ArrayList<Student>();
```

Scope **Class** **object** **Constructor**

```
private Student student = new Student();
```

Generics

What is a generic class

Collections such as the ArrayList are an example of a generic class (also known as a ‘parameterized class’).

These generic classes utilise the ‘diamond notation’ **< > and substitute the placeholders in the class definition for the type placed in the < >**

This reduces duplication as there is no need to create separate classes or methods which only work with one type.

Portion of the ArrayList class

```
/**
*   A portion of the ArrayList generic class
*   @param <E> e short for element
*/
public class ArrayList<E>…
{
    public boolean add(E e)
    {
      …
    }

    public E remove(int index)
    {
       …
    }
}
```

Full documentation available at:
https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Add to an ArrayList

Adding objects through the method

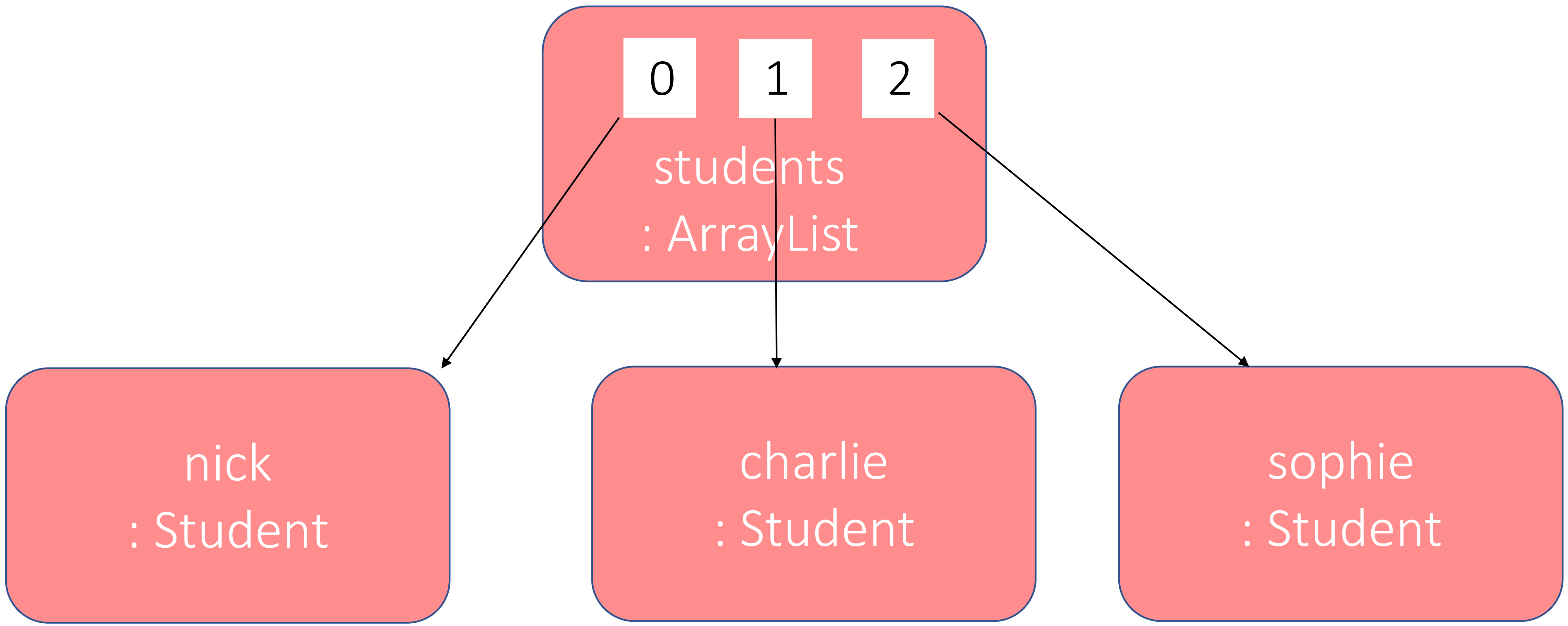
Can call the ‘add’ method through an ArrayList object

students.**add**(nick)

students.**add**(charlie)

students.**add**(sophie)

Adding an item to an ArrayList



Iterating through an ArrayList (for each)

for each loop with collection

The for each loop can be used to iterate through collections of objects.

Requires an object to be declared of the type of item that is in the collection:

```
for(Student student : students)
{
    student.print();
}
```

for each loop with collection

The for each loop can be used to iterate through collections of objects.

Requires an object to be declared of the type of item that is in the collection:

Class **object** **ArrayList**

```
for(Student student : students)
{
    student.print();
}
```

call print on each object in the ArrayList

Finding an item in an ArrayList

Finding an item in an ArrayList

We can take the same for each loop and use to check each object individually…

```
public Student findByID(int id)
{
    for(Student student : students)
    {
        if(student.getID() == id)
            return student;
    }
    return null;
}
```

Finding an item in an ArrayList

… and can check to see whether the value we are searching for matches a value in an item of the ArrayList

```
public Student findByID(int id)
{
    for(Student student : students)
    {
       if(student.getID() == id)
           return student;
    }
    return null;
}
```

Finding an item in an ArrayList

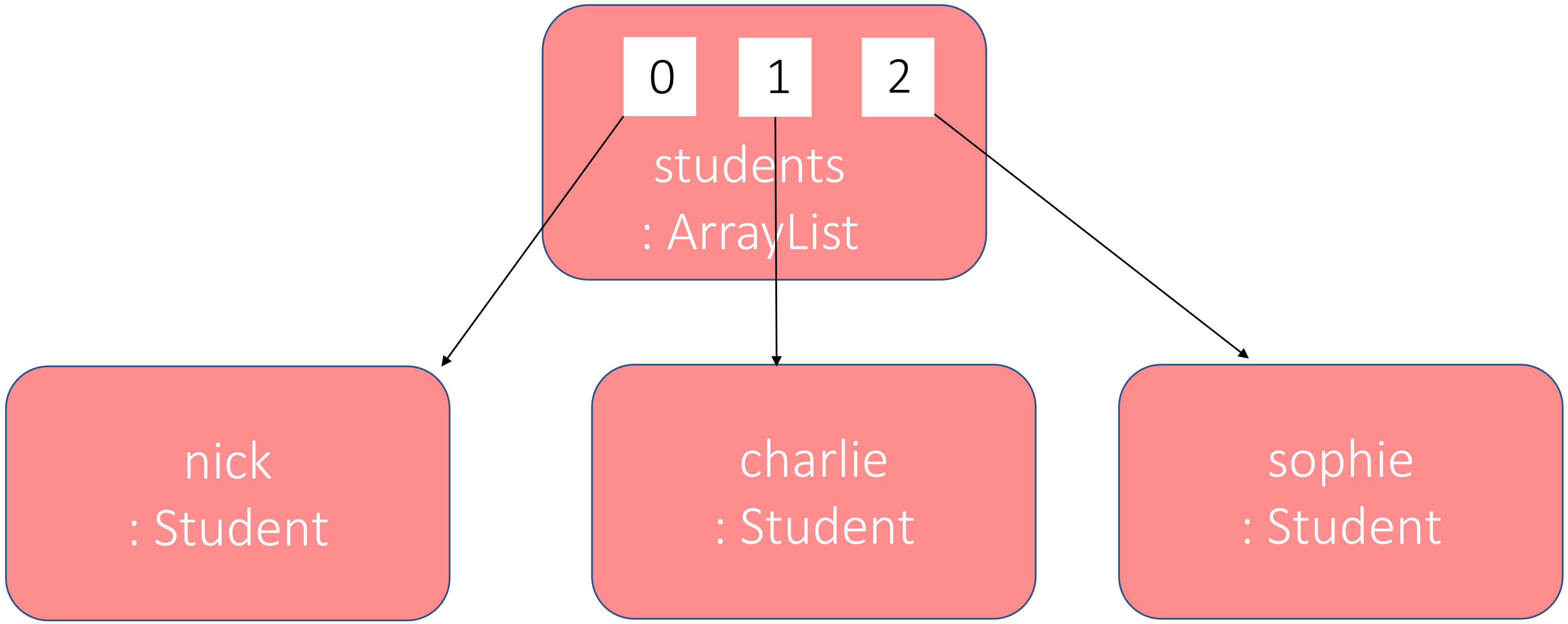
… and can check to see whether the value we are searching for matches a value in an item of the ArrayList

```
public Student findByID(int id)
{
    for(Student student : students)
    {
        if(student.getID() == id)
            return student;
    }
    return null;
}
```

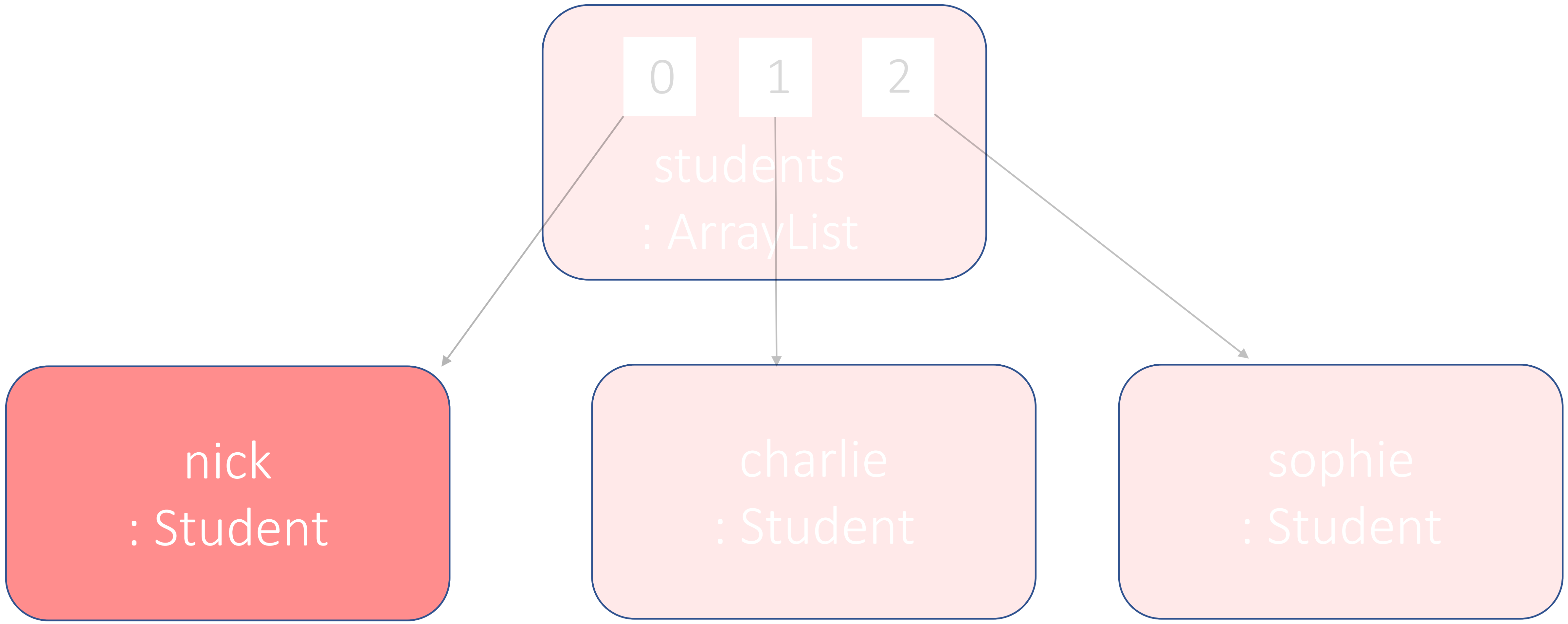
Example

Let’s say that we are looking to **return the student object that has the id value of 4231**

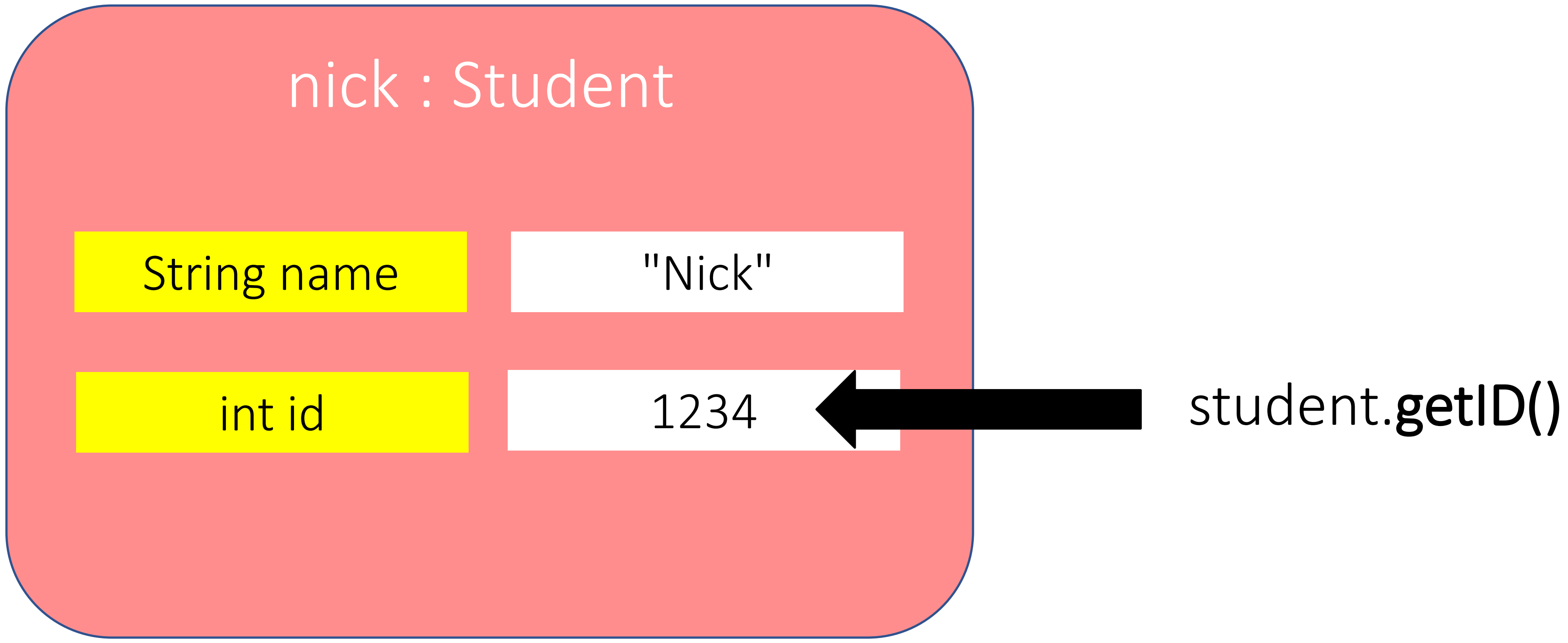
Check each item in the ArrayList



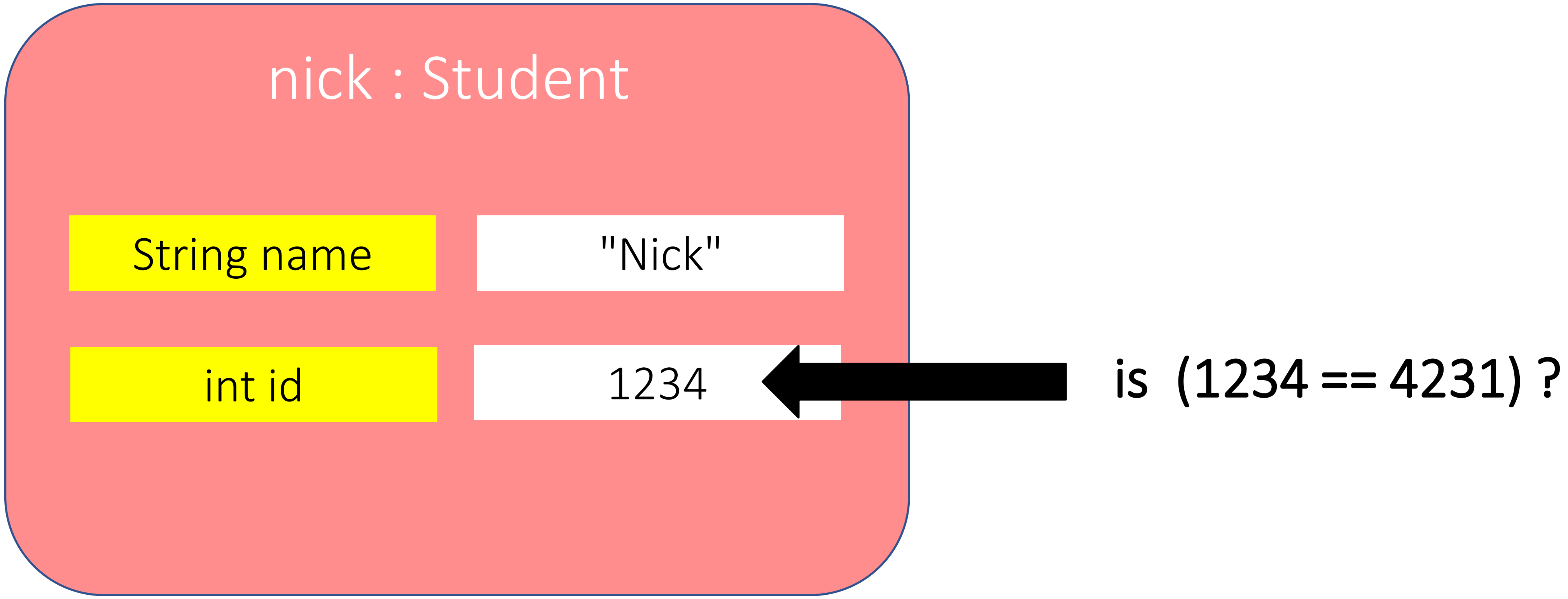
Check the first item



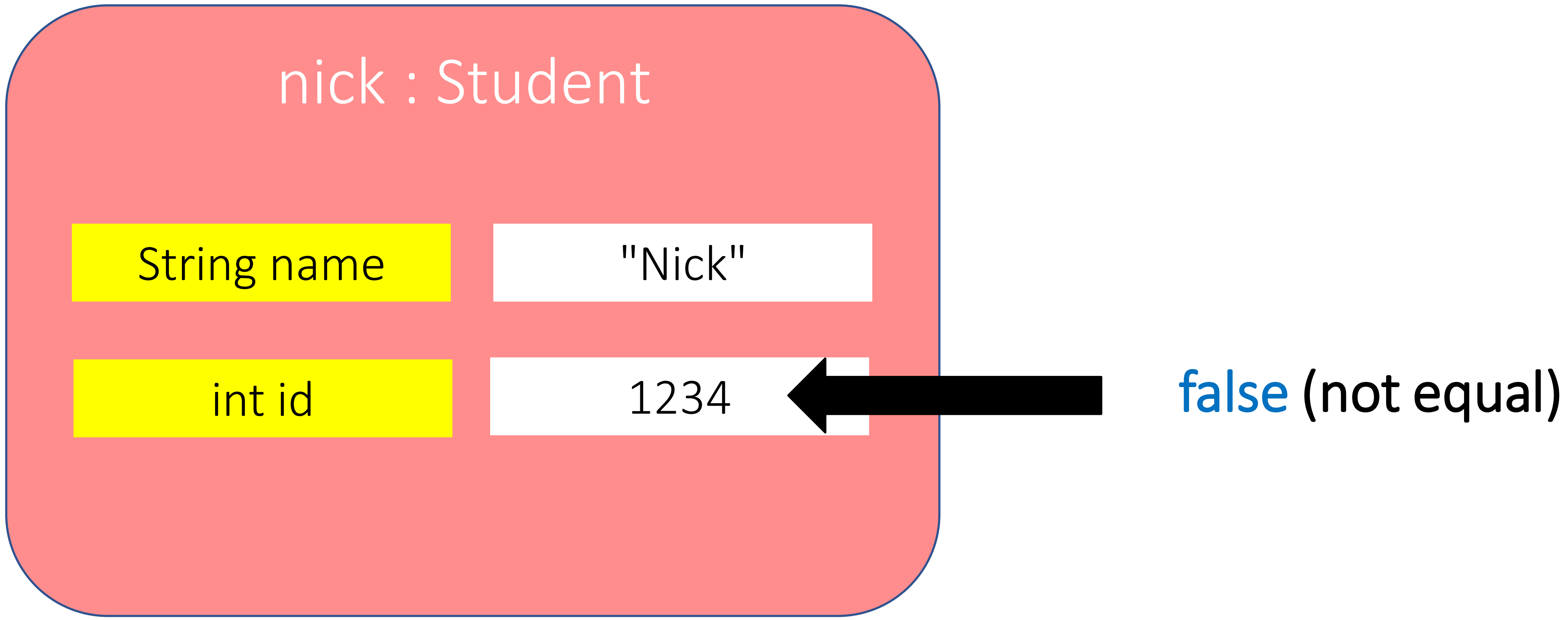
Return the id



Check the id



Not equal so move onto next item

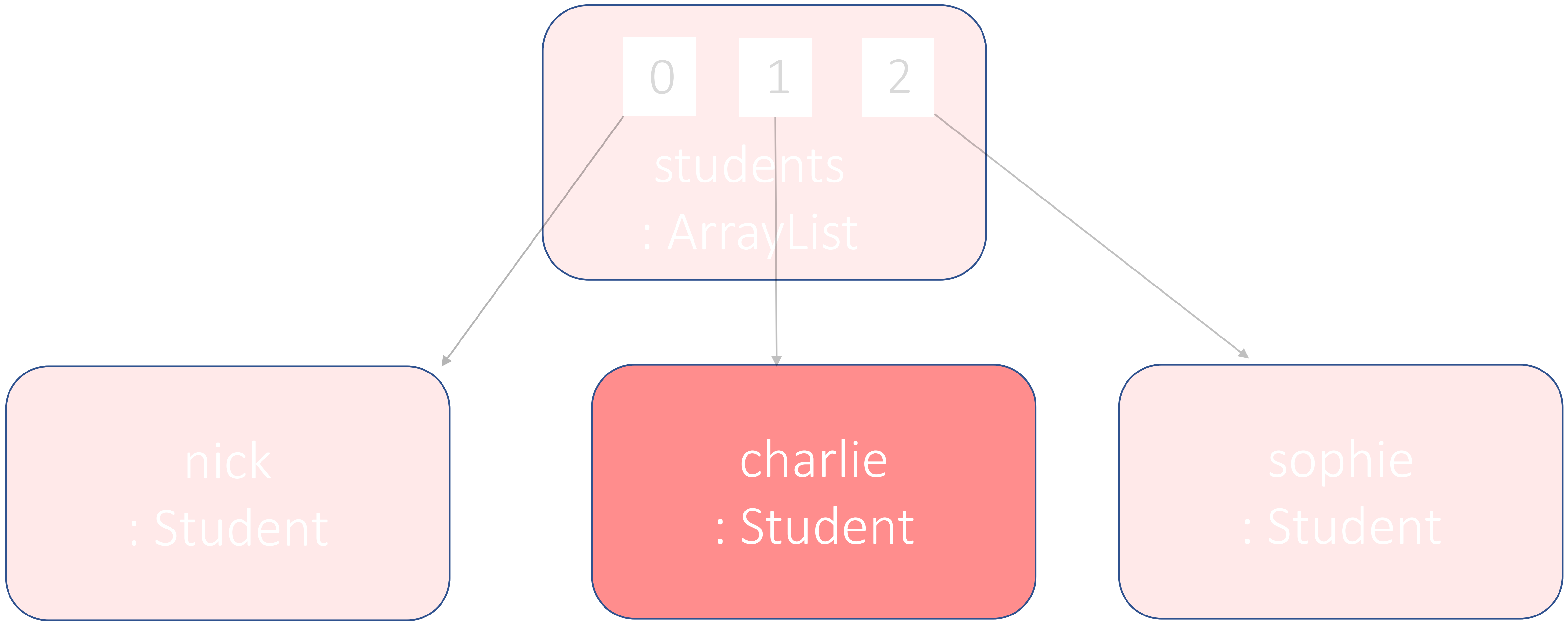


Reminder about the for each loop

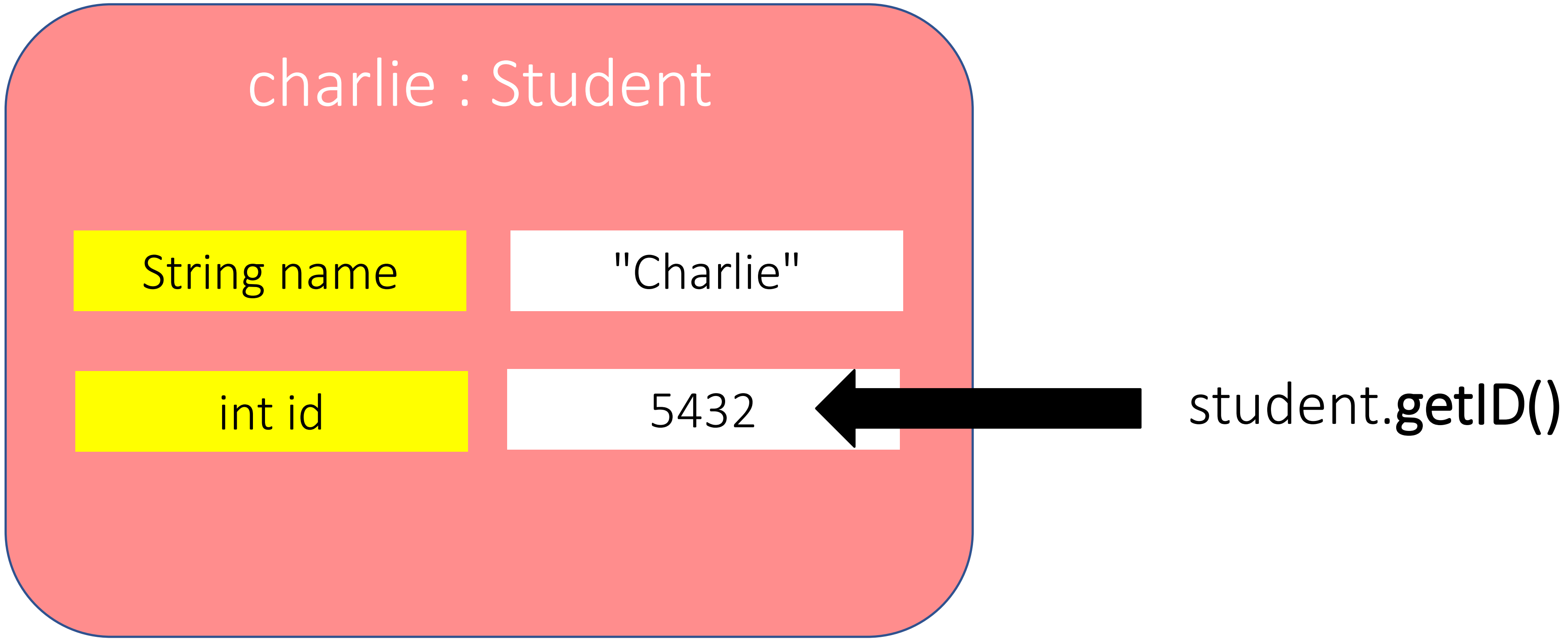
… and can check to see whether the value we are searching for matches a value in an item of the ArrayList

```
public Student findByID(int id)
{
    for(Student student : students)
    {
        if(student.getID() == id)
            return student;
    }
    return null;
}
```

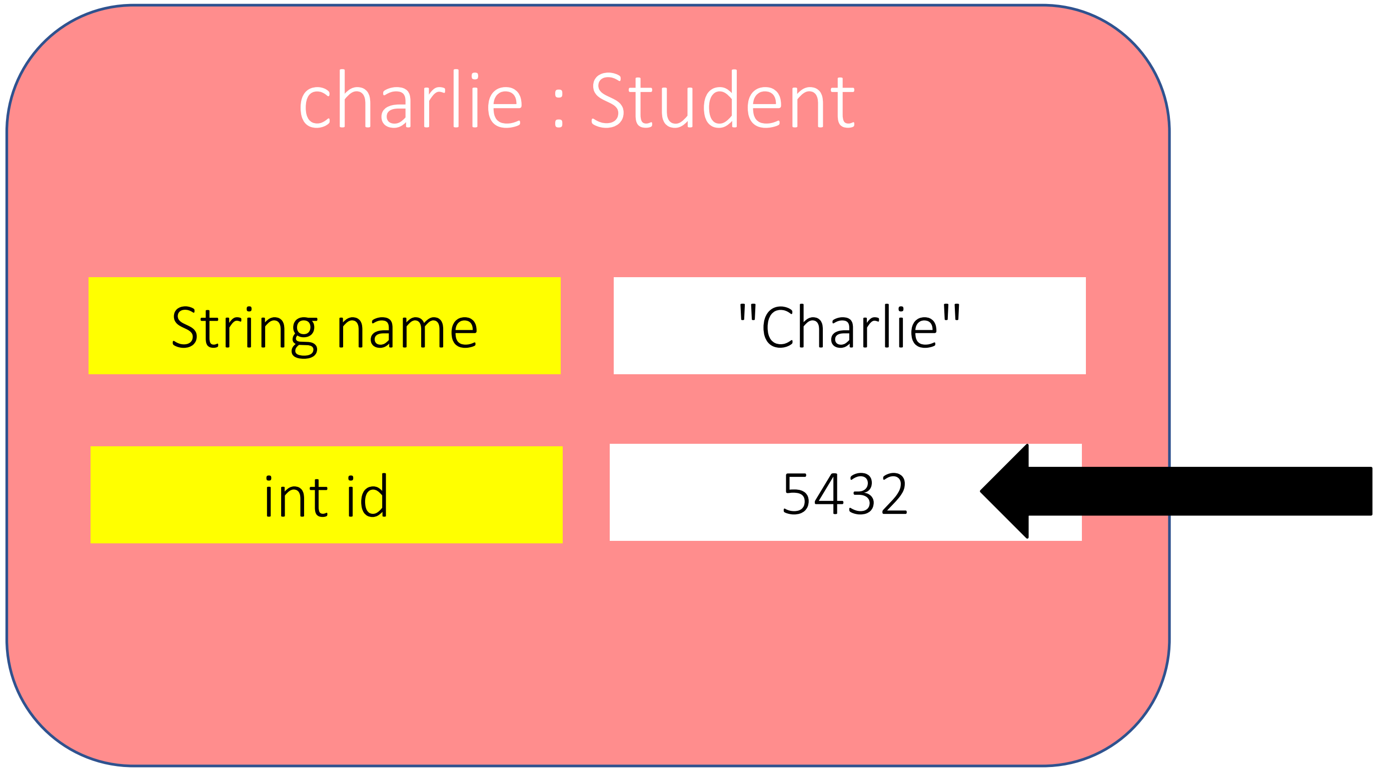
Check each item in the ArrayList



Return the id

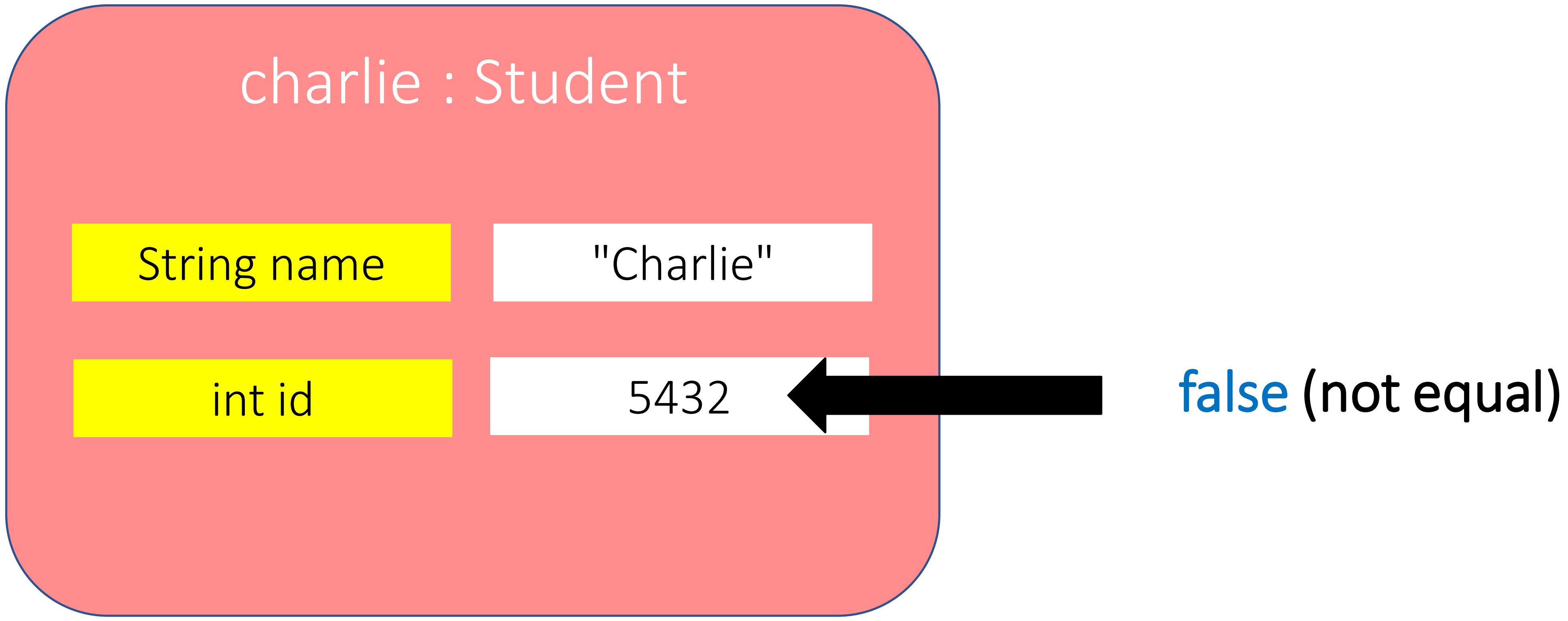


Check the id

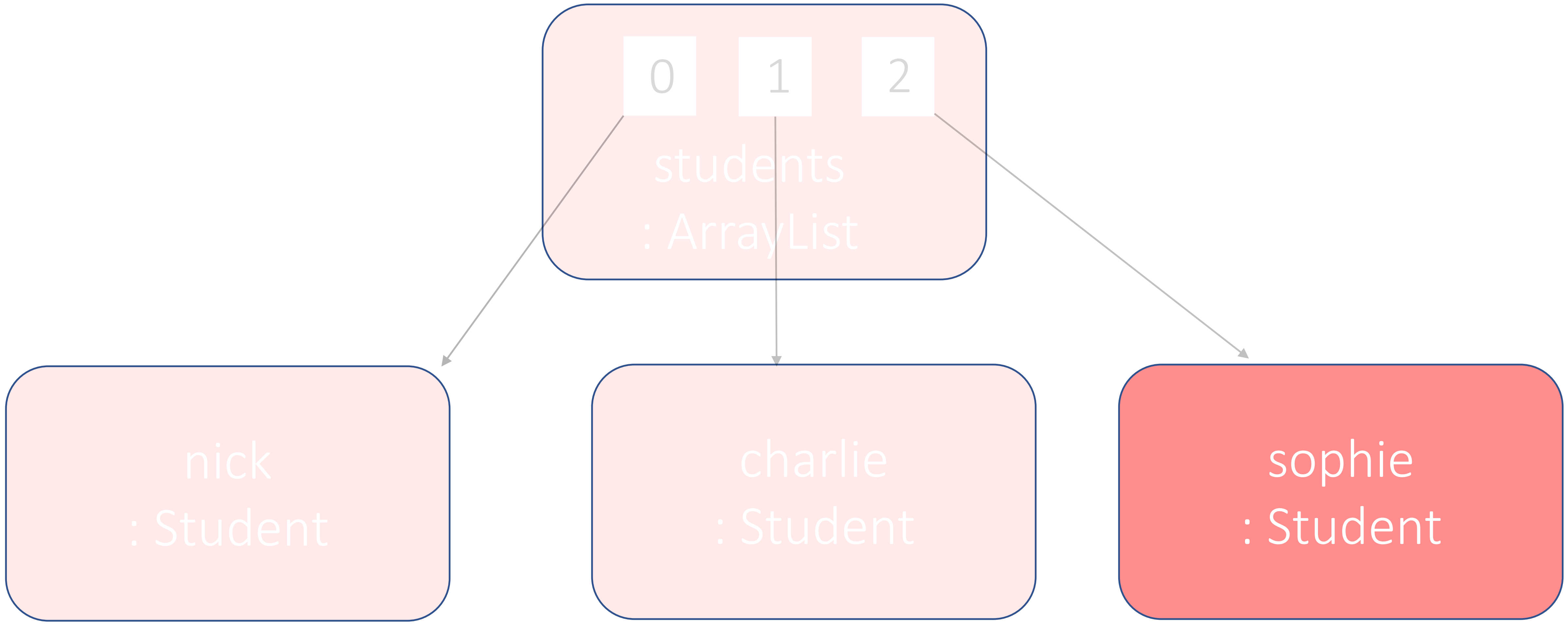


is (5432 == 4231) ?

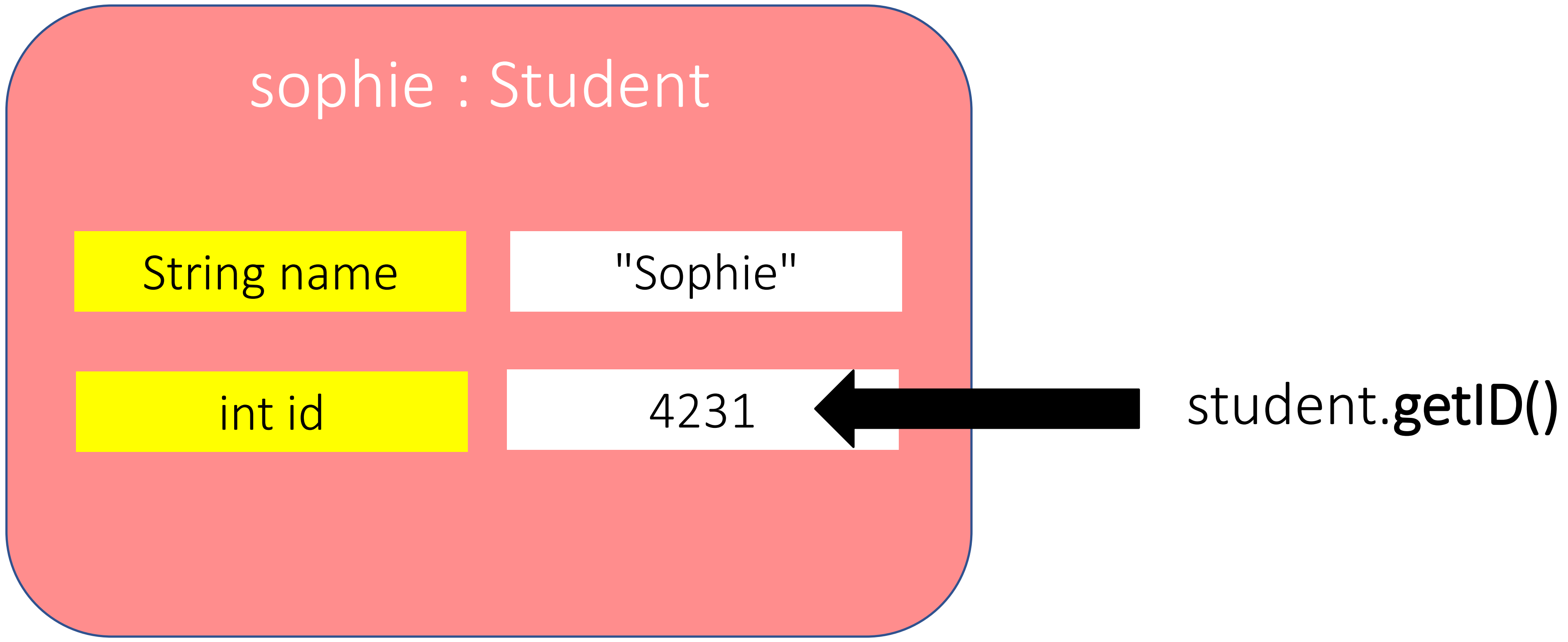
Not equal so move on to next item



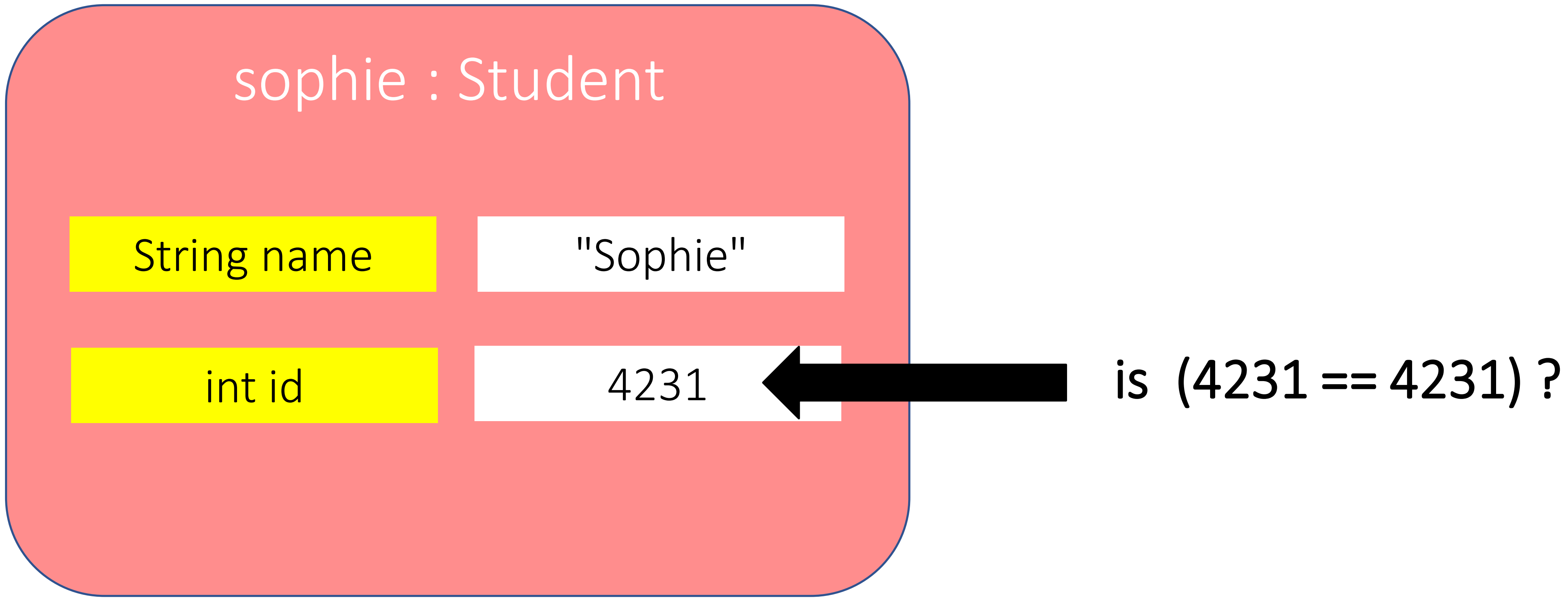
Check each item in the ArrayList



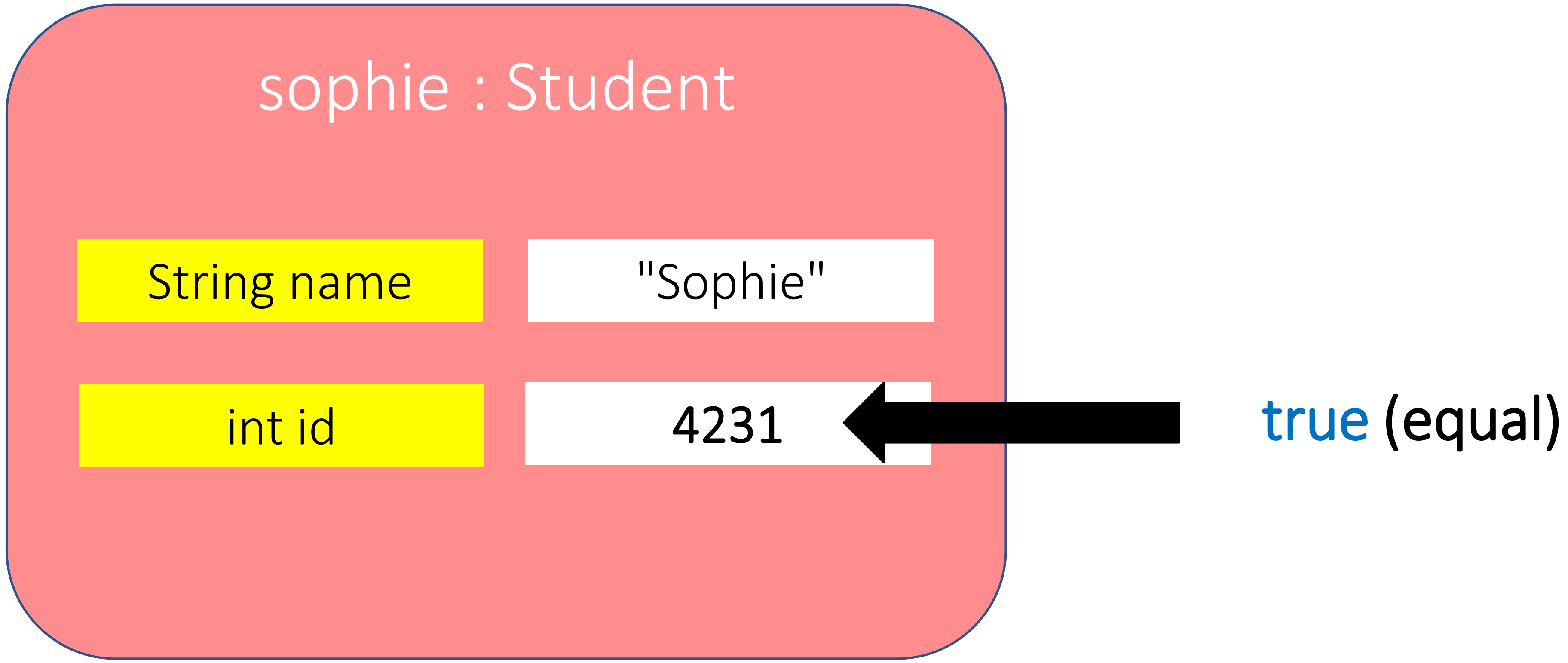
Return the id



Check the id



Match! So return object



Return the object ‘found’

Once an object with a value that matches the sought value is found, return that object and end the search

```
public Student findByID(int id)
{
    for(Student student : students)
    {
        if(student.getID() == id)
            return student;
    }
    return null;
}
```

Removing from an ArrayList

Removing an item in an ArrayList

Step 1 – **find** the object we want to remove

Step 2 – **remove** the located object (if found)

```
public Student remove(int id)
{
    Student student = findByID(id);
    if(student != null)
        students.remove(student);
    else
        System.out.println("Could not find student");
}
```