جامعة الملك سعود
King Saud University
1957
كلية علوم الحاسب والمعلومات
قسم علوم الحاسب

Arrays of objects

CSC 113: Computer programming II

Arrays of objects

- In Java, in addition to arrays of primitive data types, we can declare arrays of objects
- An array of primitive data is a powerful tool, but an array of objects is even more powerful.
- The use of an array of objects allows us to model the application more cleanly and logically.

Arrays of objects

We will use Student class to illustrate the use of an array of objects.

```
public class Student
{           private String name;
            private int age;
            private char gender;
            public Student ()   {  age=0; name=" ";   gender=' '; }
            public Student (String na, int ag, char gen)  {
                        setAge(ag);   setName(na);   setGender(gen);
            }
            public Student (Student st)      { setstudent (st);}
            public void setStudent ( Student s)      {
                        age=s.age; gender =s.gender;
                        name=s.name. substring(0, s.name.length());
            }
            public void setAge (int a)                { age=a;  }
            public void setGender (char g)            { gender=g;  }
            public void setName(String na)            { name= new String(na);  }
            public int getAge()                       { return age;  }
            public char getGender ()                  { return gender;  }
            public String getName ()                  { return name;  }
}
```

Arrays of objects: Creation

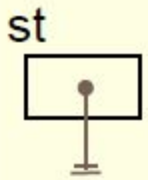
Code

A

```
Student[ ]  st;
st = new Student[20];
st[0] = new Student( );
```

Only the name pr is declared, no array is allocated yet.

State of Memory



After A is executed

Arrays of objects: Creation

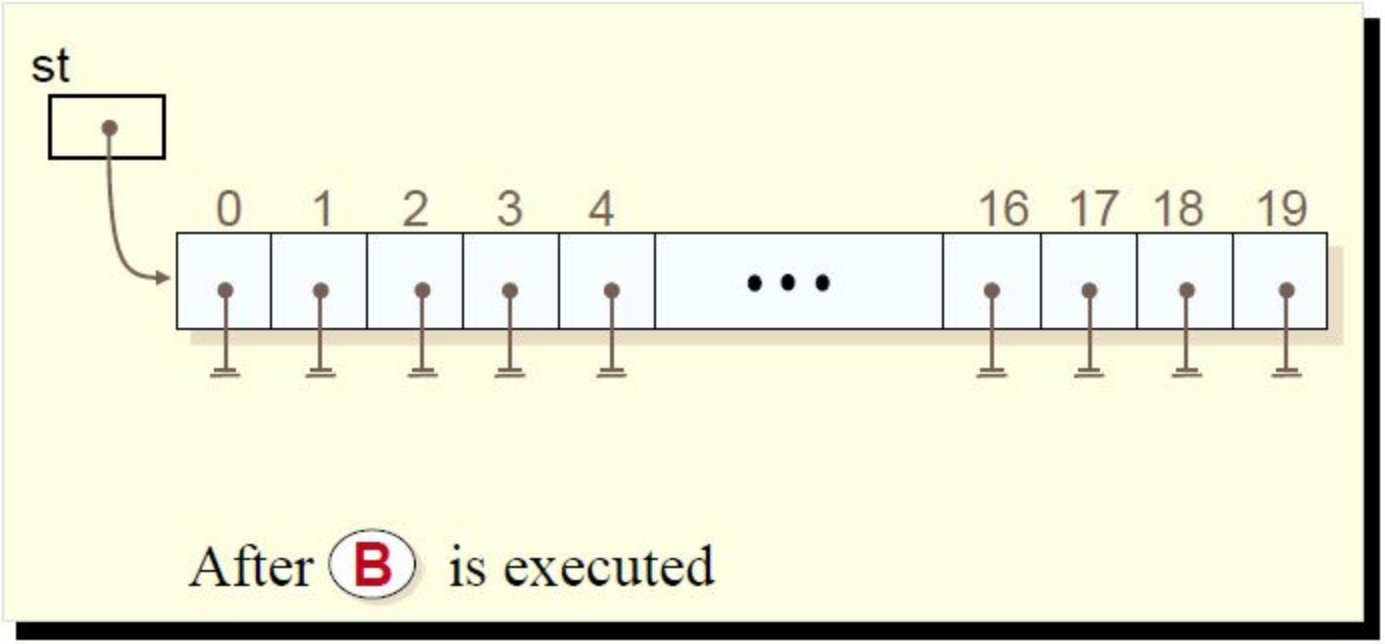
Code

B

```
Student[ ]  st;
st = new Student[20];
st[0] = new Student( );
```

Now the array for storing 20 Student objects is created, but the Student objects themselves are not yet created.

State of Memory



Arrays of objects: Creation

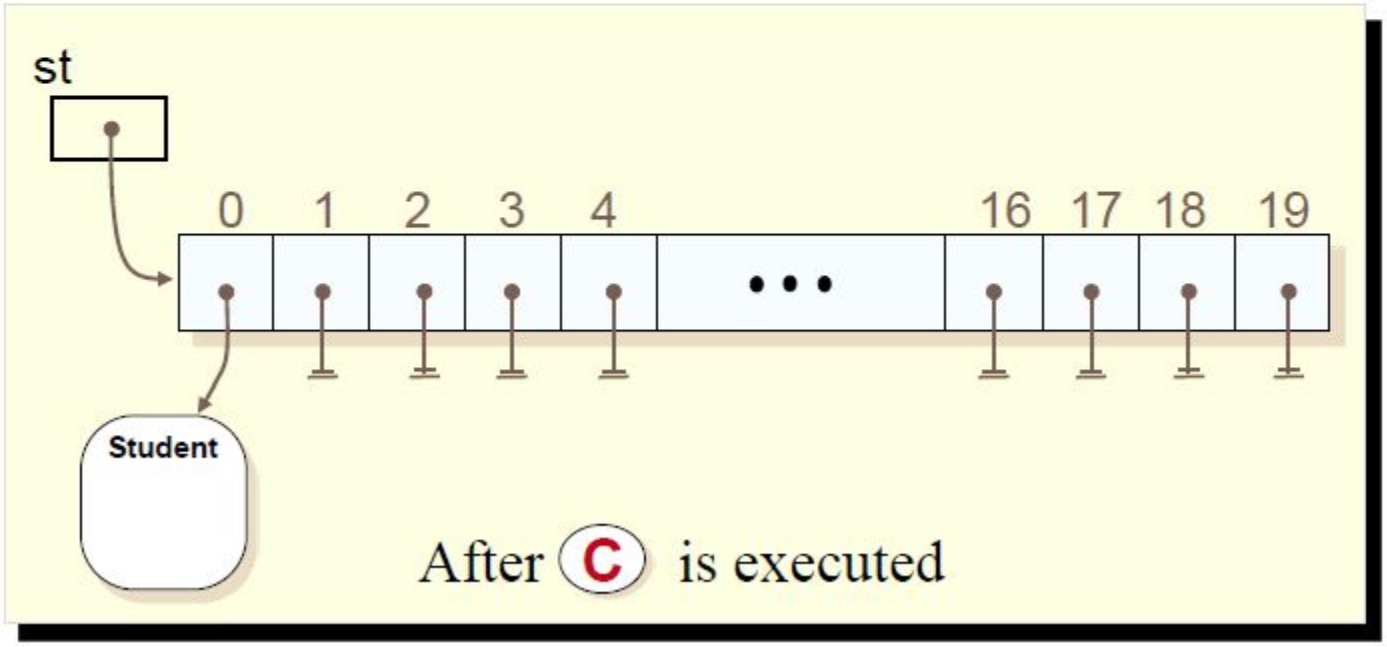
Code

```
Student[ ]  st;
st = new Student[20];
st[0] = new Student( );
```

C

One Student object is created and the reference to this object is placed in position 0.

State of Memory

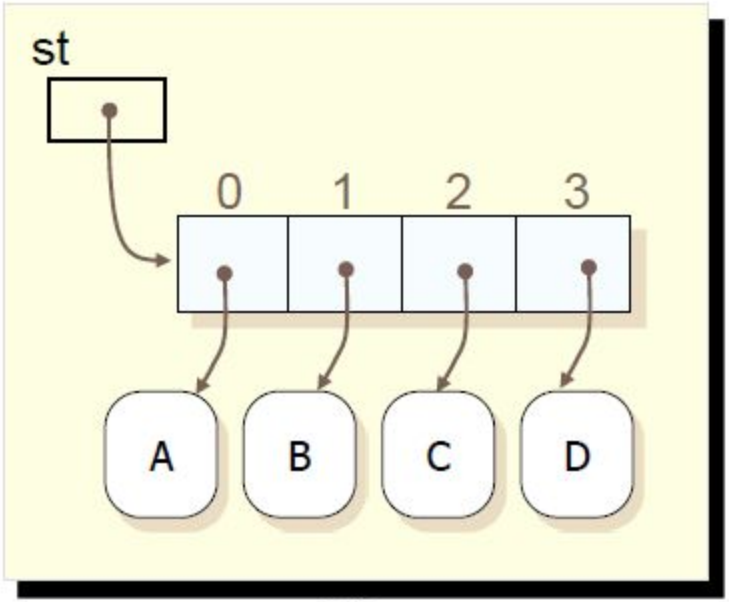


Arrays of objects: Deletion (method 1)

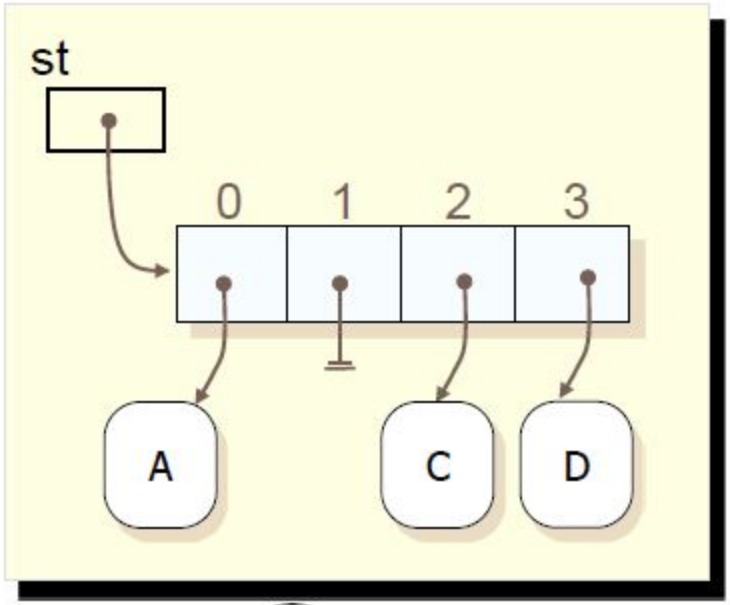
A

```
int Idx = 1;
st[Idx] = null;
```

Delete Student B by setting the reference in position 1 to null.



Before A is executed



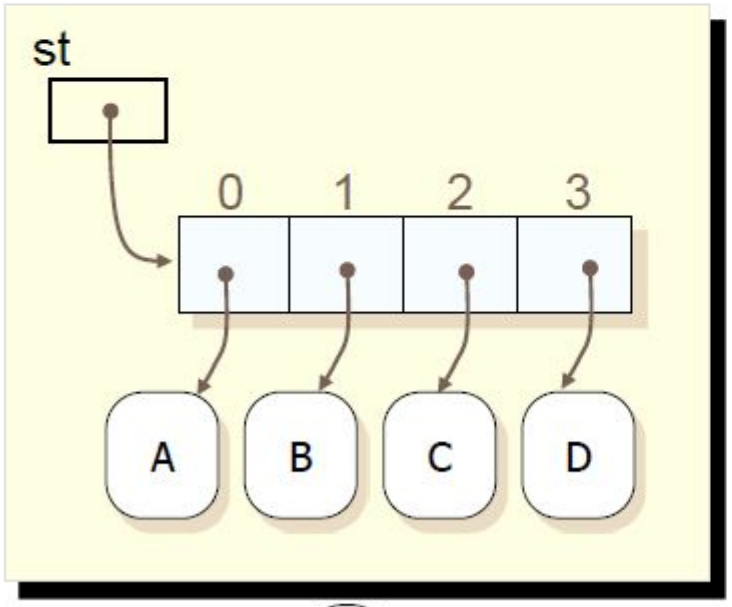
After A is executed

Arrays of objects: Deletion (method 2)

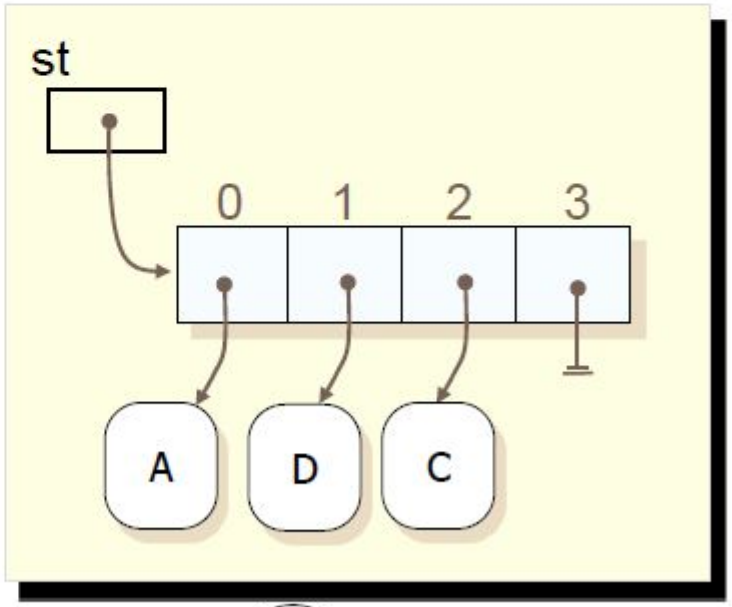
A

```
int Idx = 1, last = 3;
st[Idx] = st[last];
st[last]     = null;
```

Delete Student B by setting the reference in position 1 to the last object in the array.



Before A is executed



After A is executed

Student Array Processing – Sample 1

Create Student objects and set up the p array

Constructors

```
import java.util.Scanner;

public class Section        {
    private Student p[];
    private int nbs;
    Scanner input = new Scanner(System.in);

    public Section( int size )  {
            p = new Student[size];
            nbs = 0;
    }

    public Section( Student pr[ ] ) {
        p = new Student[pr.length];
        for (int i =0; I < p.length;  i++) {
            p[i]= new Student( pr[i] ); //---p[i]=pr[i] ---
           nbs ++; }
    }
```

Setters

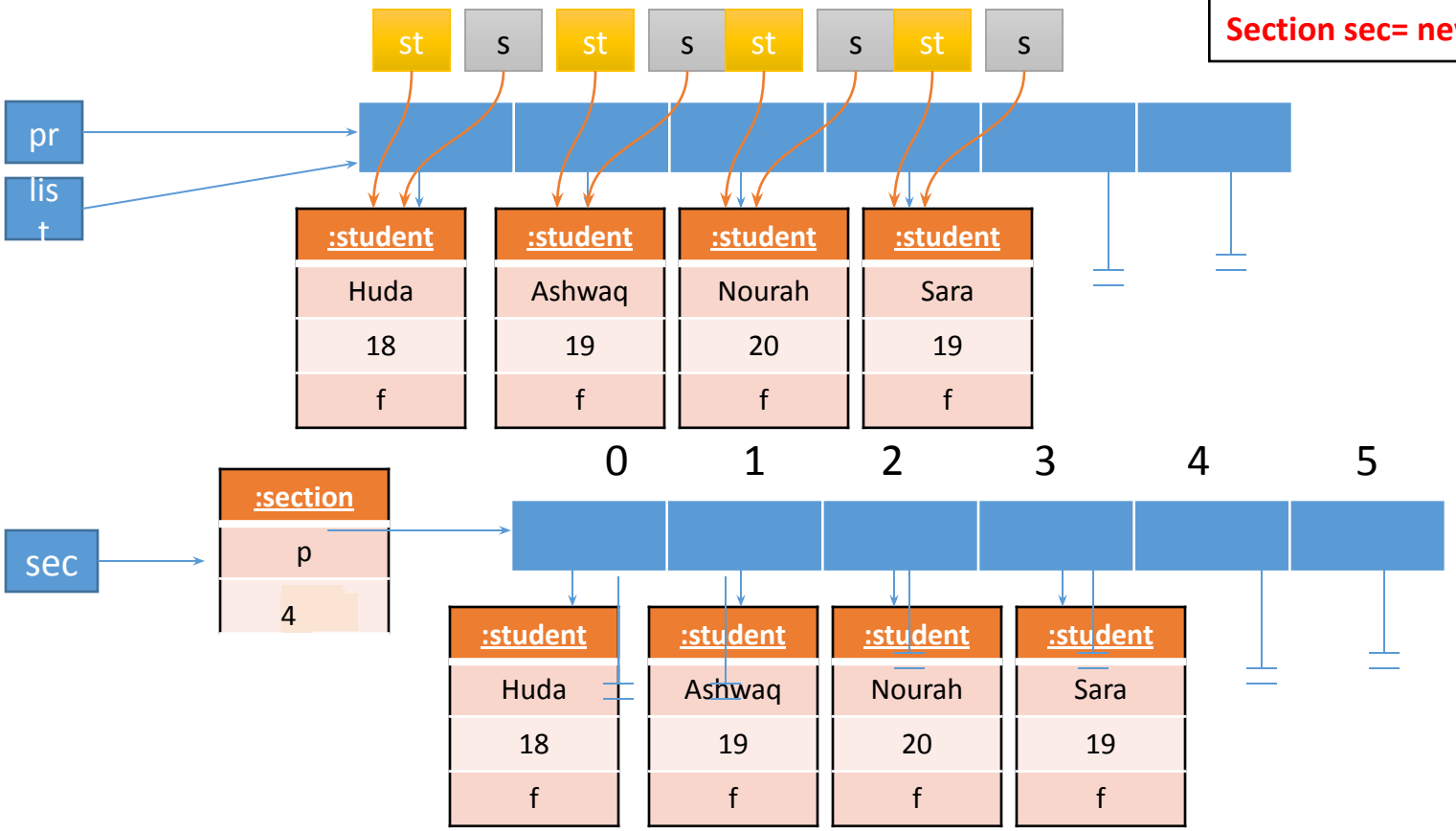
```
public void setSectionStudents ( Student [ ] pr ) {
    for (int i =0; (i< pr.length) && (i <p.length); i++)
    {    P[i]=new Student();
        p[i].setStudent( pr[i] );
        nbs++;
    }
}

public void setSectionStudents ( )        {
    String s = "";
    for (int i =0; i< p.length; i++) {
        p[i] = new Student();
        p[i].setName( input.next() + input.nextLine() );
        p[i].setAge( input.nextInt() );
        s = input.next( );
        p[i].setGender( s.charAt(0) );
    }
    nbs = p.length;
}
```

Constructor Section (Student pr[])

```
public Section( Student pr[ ] ) {
    p = new Student[pr.length];
    for (int i =0; I < p.length;  i++) {
        p[i]= new Student( pr[i] ); //---p[i]=pr[i] -
         nbs ++; }
}
```

```
public Student (Student st)      { setstudent (st);}
public void setStudent ( Student s)      {
            age=s.age; gender =s.gender;
            name=s.name. substring(0, s.name.length());
}
```

Main:
Section sec= new Section(list);



Student Array Processing – Sample 1

Add an object, parse all the array cells

Adding an object to the array

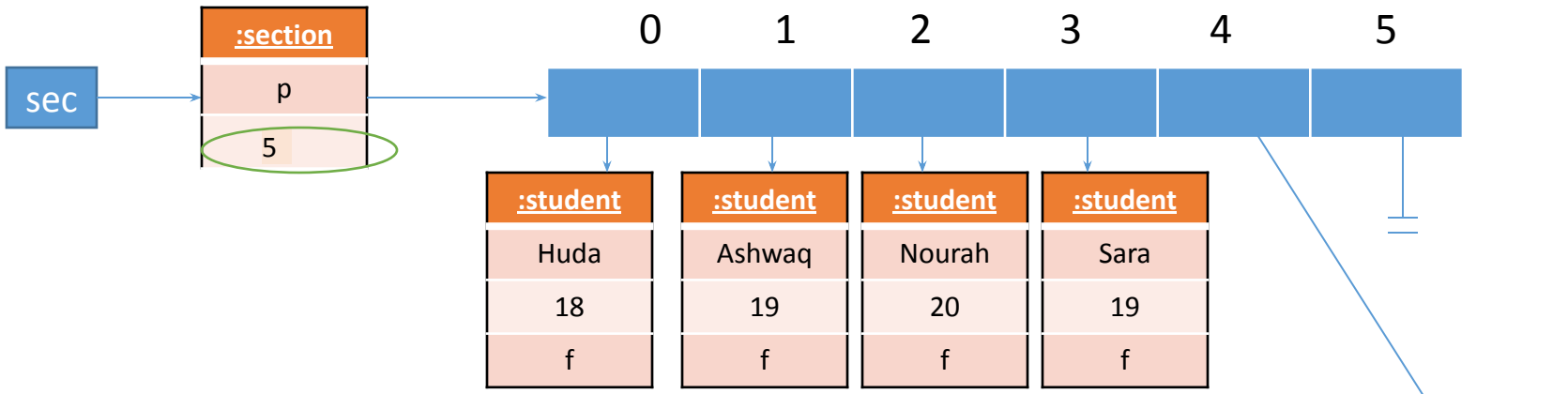
```
public boolean addStudent( Student p1 )
{
    if (nbs = = p.length)
        return false;
    p[nbs++] = p1;  // -- p[nbs] = p1; nbs++ --
    return true;
}
```

Going through all the array cells

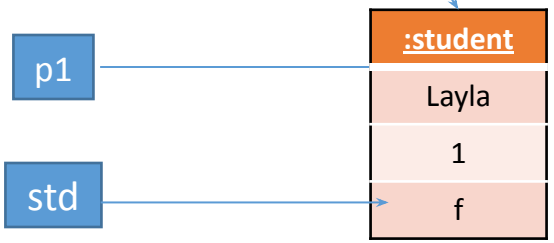
```
 //--- Average of all ages -----
public double averageOfAge( )
{
    double s=0.0;
    for(int i =0; i<=nbs-1; i++)
        s+=p[i].getAge( );
    return ( s/nbs );
}
```

Add an object

Main:
sec .addStudent(std)




```
public boolean addStudent( Student p1 )
{
    if (nbs = = p.length)
        return false;
    p[nbs++] = p1;  // -- p[nbs] = p1; nbs++ --
    return true;
}
```



Student Array Processing – Sample 1

Finding an object in an array

Find a student by 1 parameter

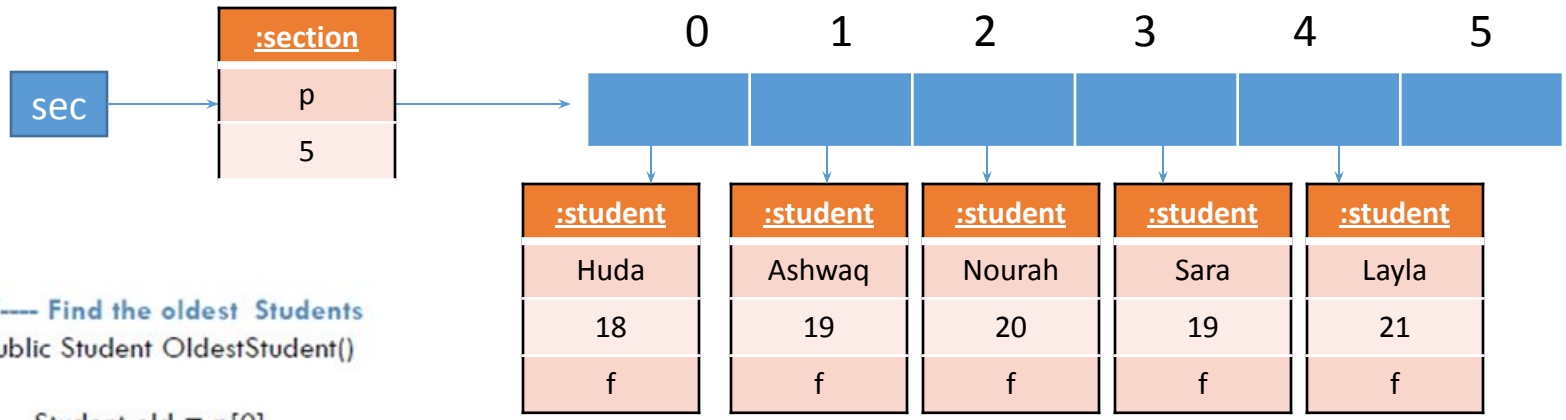
```
//---- Find the oldest  Students
public Student OldestStudent()
{
      Student old = p[0];
      for (int i =1; i<=nbs-1; i++)
            if (old.getAge() < p[i].getAge())
                              old =p[i];
      return (old);
}
//--- search  for a particular Student ----
public boolean findStudentByName(String na)
{
      for (int i=0; i<nbs; i++) {
            if (p[i].getName().equals(na)== true)
                  return (true);
      }
      return (false);
}
```

Find a student by all its parameters

```
//---- return index of a Student if exist and -1 if
     not
  public int findStudent(Student pr)
  {
       for (int i=0; i<nbs; i++) {
         if (p[i].getName().equals(pr.getName())==
    true)
              if (p[i].getAge() == pr.getAge())
                if (p[i].getGender()== pr.getGender())
                  return (i);

         }
       return (-1);
  }
```

Find student by 1 parameter

sec .oldestStudent()




```
//---- Find the oldest  Students
public Student OldestStudent()
{
      Student old = p[0];
      for (int i =1; i<=nbs-1; i++)
            if (old.getAge() < p[i].getAge())
                        old =p[i];
      return (old);
}
```

Student Array Processing – Sample 1

Deleting an object from the array

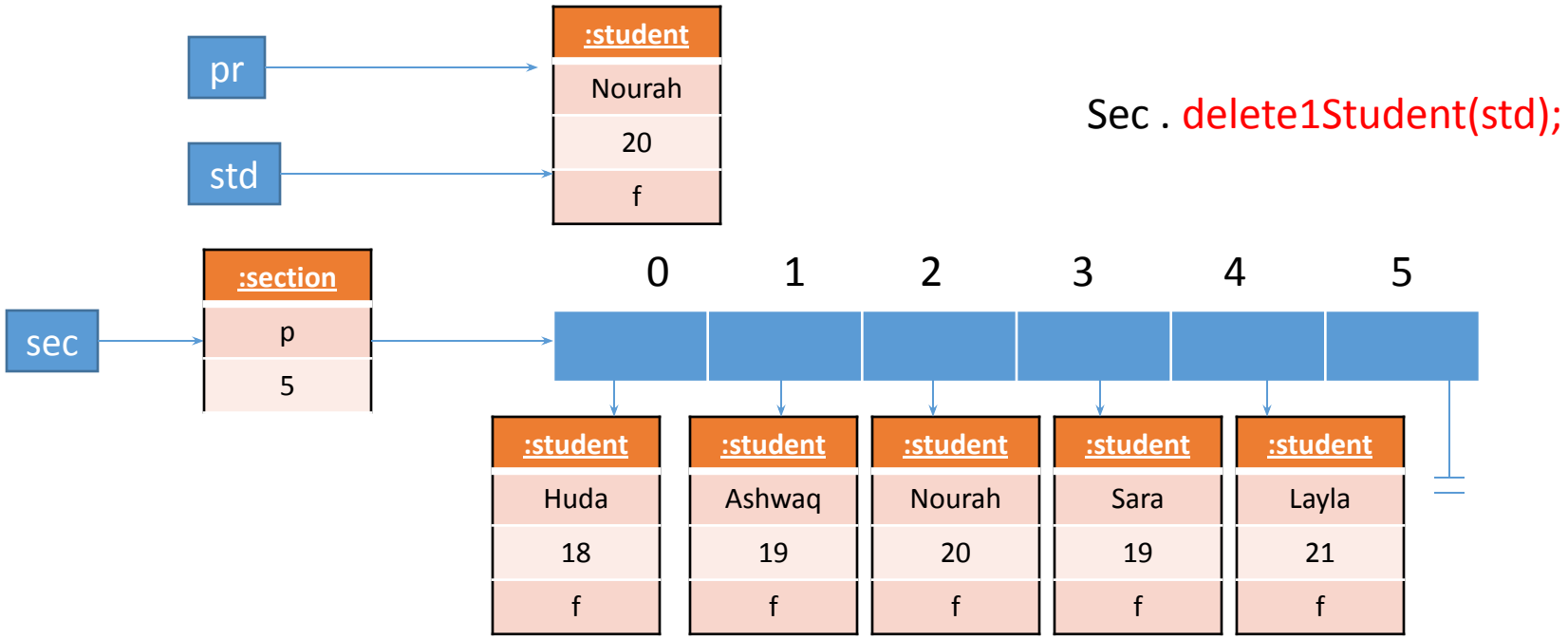
Simple object deletion

```
public  boolean delete1Student(Student pr)
{
     int x = findStudent(pr)
     if (x != -1)
     {
          p[x] = p[nbs – 1];
          p[--nbs] = null;
          return true;
     }
     return false;
}
```

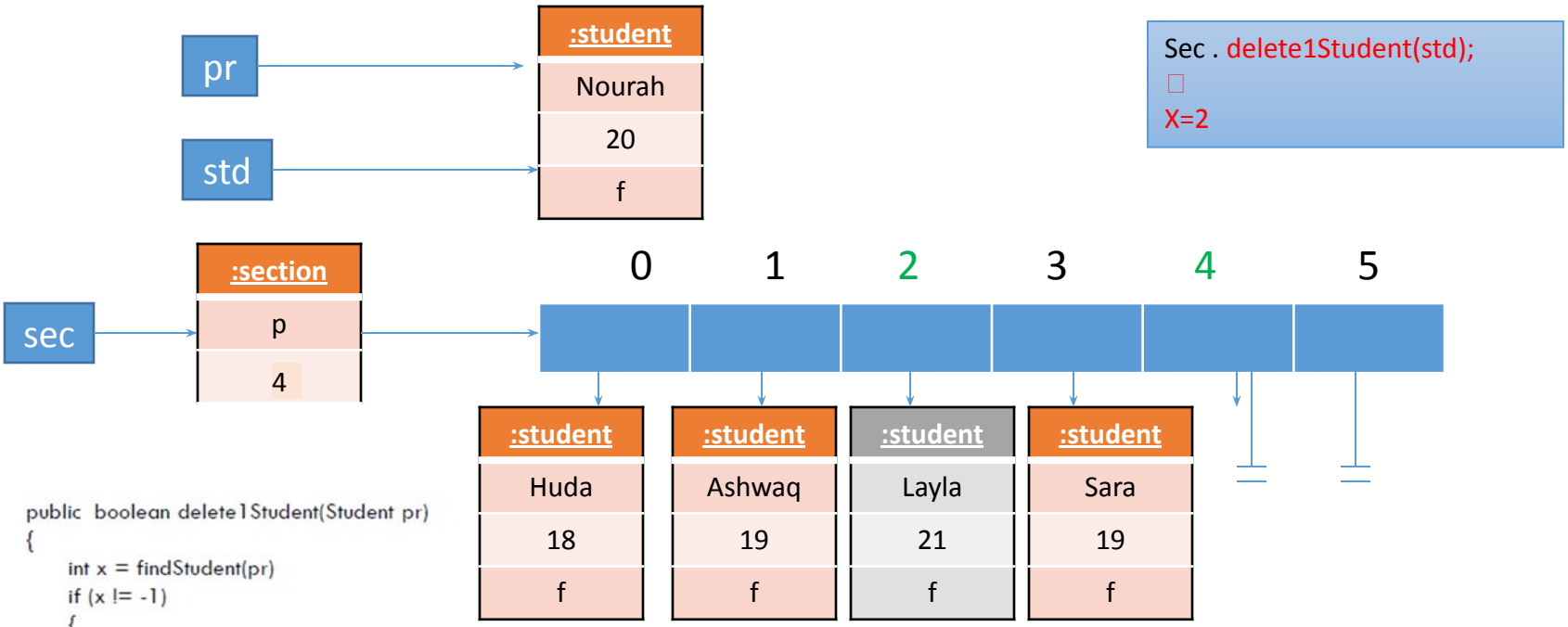
Object deletion with replacement

```
     public  boolean delete2Student(Student pr)
     {
          int x = findStudent(pr)
          if (x != -1)        {
               for (int i = x; i<nbs - 1; i++)
               p[i] = p[i + 1];
               p[--nbs] = null;
               return true;
          }
          return false;
     }
}
```

Simple object deletion

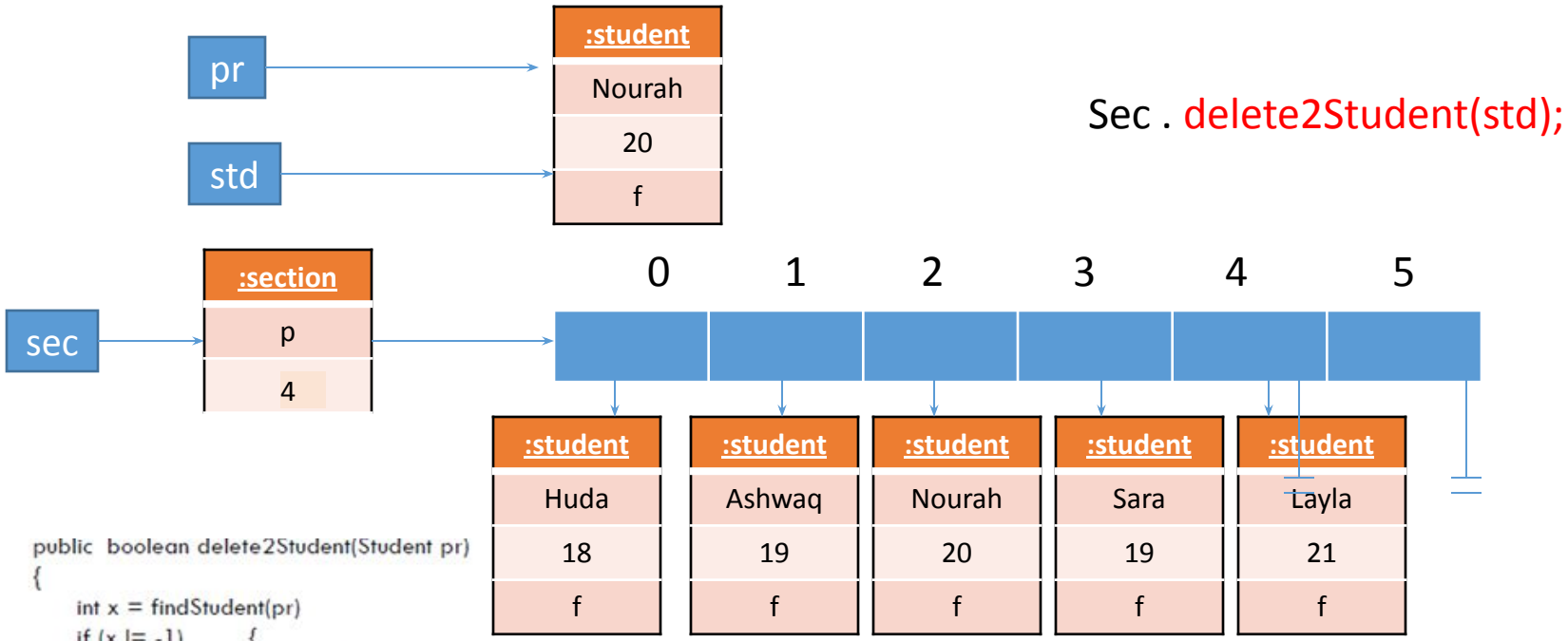


Simple object deletion




```
public  boolean delete1Student(Student pr)
{
     int x = findStudent(pr)
     if (x != -1)
     {
          p[x] = p[nbs – 1];
          p[--nbs] = null;
          return true;
     }
     return false;
}
```

object deletion with replacement




```
public  boolean delete2Student(Student pr)
{
    int x = findStudent(pr)
    if (x != -1)        {
        for (int i = x; i<nbs - 1; i++)
        p[i] = p[i + 1];
        p[--nbs] = null;
        return true;
    }
    return false;
}
}
```