Saving an Array to a File for later Loading

- Saving & loading an array is similar to saving an ArrayList:
  - Step through the array, writing each value to the file, or reading each value from the file.
  - But: Must be careful with null values.
    Must know the number of items **before** we construct the array.

- Three options:

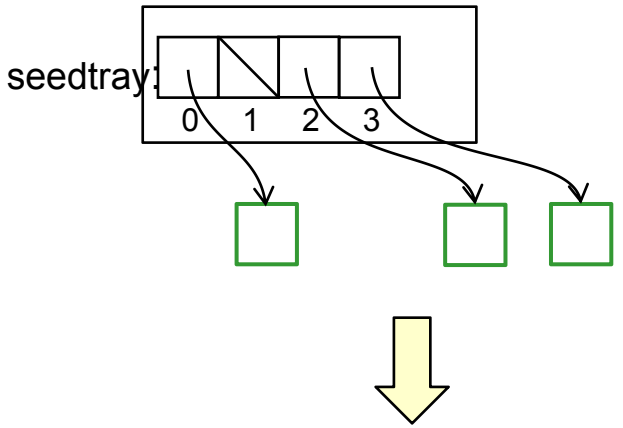
a) write one line for each item in the file; write "null" for null values
read lines into a list, then create array. Check for lines with "null"
➔ size of list tells us the size of the array,
but inefficient for very large files or arrays with lots of null values.

b) write the size of the array to the file before writing the values.
➔ can read the size, then construct array, then read the values

c) write the size of the array to the file first, then
for each non-null value, write the index and the value on a line
➔ can read the size, then construct array, then read the index – value pairs.

Saving and Loading, option (a)

```
public void save(){
    try{
        String fname = UIFileChooser.save("File for Seedtray");
        PrintStream out = new PrintStream(fname);
      for (Flower flower  :  this.seedtray) {
          if (flower == null) {
                out.println("null");
          }
          else {
                out.println(flower);
          }
      }
        out.close();
    }catch (IOException e) { UI.println("File saving failed: "+e); }
}
```

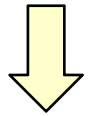


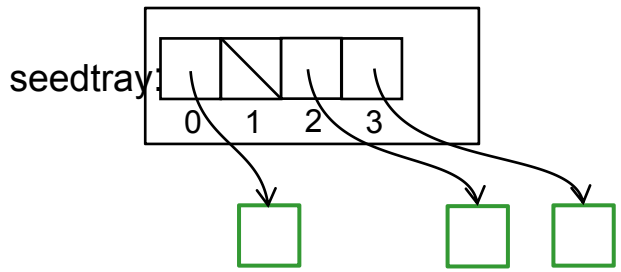

```
104 268  40  bud
null
524 245  20  picked
274  83   50  bud
                    :
```

Saving and Loading, option (a)

```
public void load(){
    try{
        String fname = UIFileChooser.open("File containing Seedtray");
        List<String> lines = Files.readAllLines(Path.of(fname));
        this.seedTray = new Flower [ lines.size() ];
      for (int i = 0; i<lines.size(); i++) {
          if ( ! lines.get(i).equals("null") ) {
              Scanner sc = new Scanner(lines.get(i));
              Flower f = new Flower(sc.nextDouble(), sc.nextDouble(), sc.nextDouble(), sc.next());
              this.seedTray [ i ] = f;
          }
      }
    } catch (IOException e) { UI.println("File loading failed: "+e); }
}
```

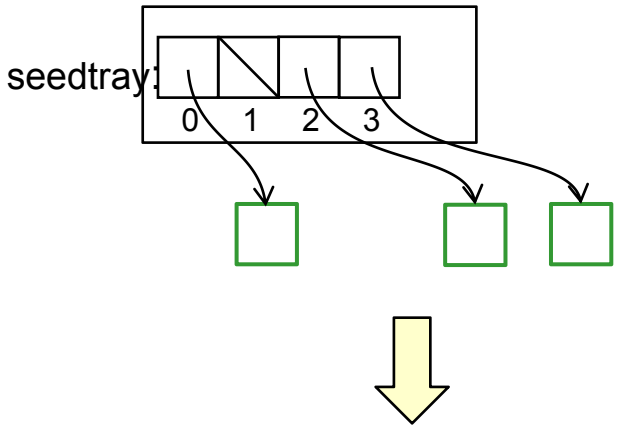
```
104 268  40  bud
null
524 245  20  picked
274  83  50  bud
                    :
```





Saving and Loading, option (b)

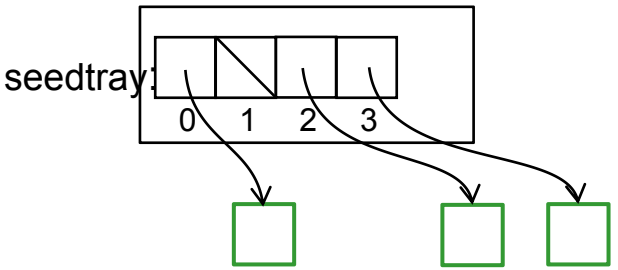
```
public void save(){
    try{
        String fname = UIFileChooser.save("File for Seedtray");
        PrintStream out = new PrintStream(fname);
        out.println(this.seedTray.length);
      for (Flower flower  :  this.seedtray) {
          if (flower == null) {
              out.println("null");
          }
          else {
              out.println(flower);
          }
      }
        out.close();
    }catch (IOException e) { UI.println("File saving failed: "+e); }
}
```



Saving and Loading, option (b)

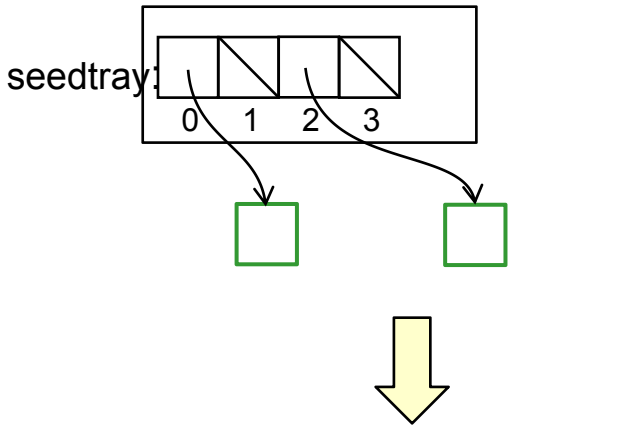
```
4
104 268  40  bud
null
524 245  20  picked
274  83  50  bud
                    :
```

```
public void load(){
    try{
        String fname = UIFileChooser.open("File containing Seedtray");
        Scanner sc = new Scanner(Path.of(fname));
        this.seedTray = new Flower [ sc.nextInt() ];
      for (int i = 0; i< this.seedTray.length; i++) {
            if (sc.hasNextDouble()) {
                Flower f = new Flower(sc.nextDouble(), sc.nextDouble(), sc.nextDouble(), sc.next());
                this.seedTray[ i ] = f;
            }
            else { sc.nextLine(); }
        }
        sc.close();
    }
    catch (IOException e){UI.println("File loading failed: "+e);}
}
```



Saving and Loading, option (c)

```
public void save(){
    try{
        String fname = UIFileChooser.save("File for Seedtray");
        PrintStream out = new PrintStream(fname);
        out.println(this.seedTray.length);
      for (int i=0; i<this.seedTray.length; i++) {
        if (this.seedTray[ i ] != null) {
            out.println( i + " " + this.seedTray[ i ] );
        }
      }
        out.close();
    }catch (IOException e) { UI.println("File saving failed: "+e); }
}
```

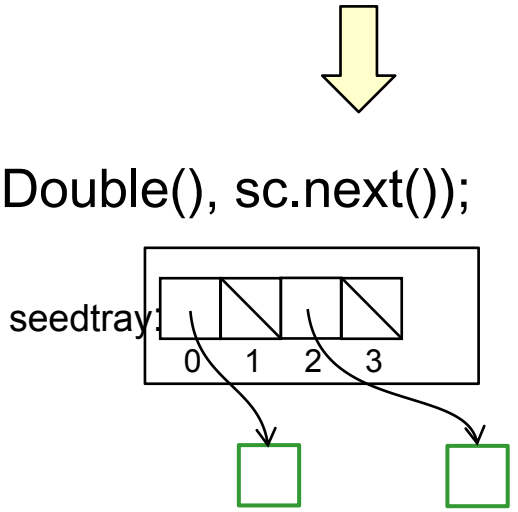



```
4
0 104 268  40  bud
2 524 245  20  picked
```

Saving and Loading, option (c)

```
4
0 104 268  40  bud
2 524 245  20  picked
```

```
public void load(){
    try{
        String fname = UIFileChooser.open("File containing Seedtray");
        Scanner sc = new Scanner(Path.of(fname));
        this.seedTray = new Flower [ sc.nextInt() ];
      while ( sc.hasNext() )  {
            int indx = sc.nextInt();
            Flower f = new Flower(sc.nextDouble(), sc.nextDouble(), sc.nextDouble(), sc.next());
            this.seedTray[indx] = f;
            }
        sc.close();
    }
    catch (IOException e){UI.println("File loading failed: "+e);}
}
```

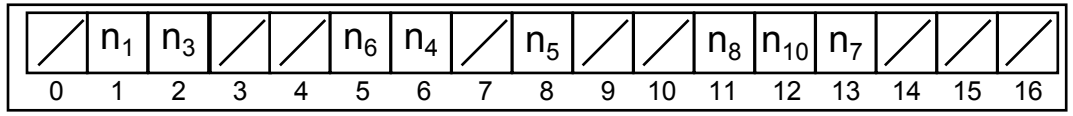


More ways of using arrays

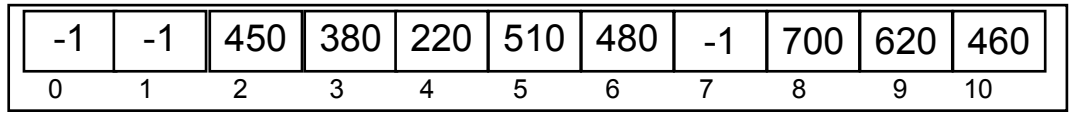
Sometimes, the index represents meaningful information:
More than just the position in the array.

For example:

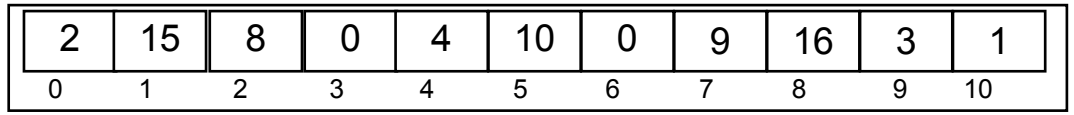
- array of guest names for hotel rooms (numbered 1 to MaxRoom)



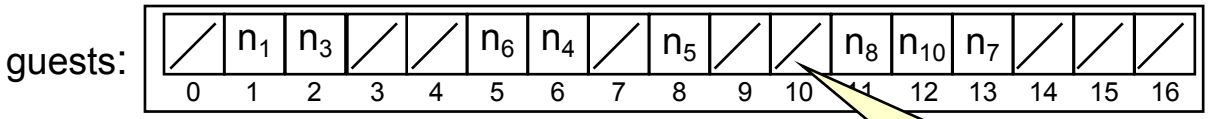
- house info for each house on street



- count of occurrences of each number



Hotel Register



null: the "not-a-real-object" value. Can always be assigned to a place of an object type

- initialise empty

```
private String[ ] guests = new String[MaxRoom+1];
  // gives indexes from 0 to MaxRoom, ignore index 0
```

- assign to room

```
public void assignGuest(String name; int room){
    this.guests[room] = name;
}
```

Don't need to step through array!

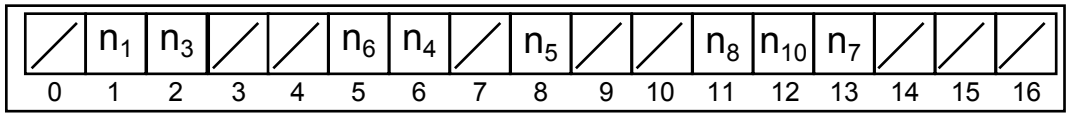
- look up to see if free

```
public boolean isFree(int room){
    return (this.guests[room]==null);
}
```

Hotel Register: remove

Checkout guest from room

guests:



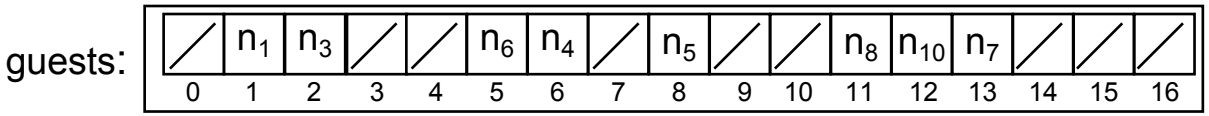

```
/* returns true if name was checked out of room successfully,
    false otherwise */
public boolean checkout(String name, int room){
     if ( this.guests[room].equals(name) ){
          this.guests[room] = null;
          return true;
     }
     return false;
}
```

Problem if null!
How do we fix it?

```
if ( this.guests[room] != null  &&  this.guests[room].equals(name) ) …
if ( name.equals(this.guests[room]) ) …
```

Alternative

Checkout guest: search and remove:




```
/* returns true if name was checked out successfully,
   false otherwise */
public boolean checkout(String name){
    for (int rm=1;   rm<this.guests.length;  rm++){      // or <=MaxRoom
        if (this.guests[rm] != null && this.guests[rm].equals(name) ) {
                this.guests[rm] = null;
                return true;
            }
        }
    }
    return false;
}
```

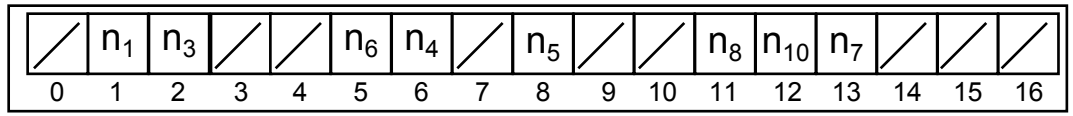
Example of "Any" pattern

Find an empty room

Find the index of an empty room (return -1 if no empty rooms)

```
public int findEmpty(){
    for (int rm=1;  rm<this.guests.length;  rm++){     // or <=MaxRoom
        if (this.guests[rm]==null)   {  return rm;  }
    }
    return -1;
}
```

guests:



Check a guest into an empty room (return room number)

```
public void checkIn(String name){
    this.guests[ this.findEmpty() ] = name;
}

public boolean checkIn(String name){
    int rm = this.findEmpty();
    if (rm < 0) { return false; }
    this.guests[rm] = name;
    return true;
}
```

What's the problem?
How do we fix it?

Arrays of Booleans

- To keep track of numbers we have seen, efficiently
- Eg: looking for duplicate numbers in a file of integers: read file, check and set

```
Scanner sc = new Scanner(new File(FileDialog.open("file to check")));
boolean[ ] numbersSeen = new boolean[100000];
while ( sc.hasNext() ){
    if ( sc.hasNextInt() ){
        int num = sc.nextInt();
        if ( num>=0 && num<100000 ){
            if (numbersSeen[num] )
                    System.out.println(num + "is a duplicate");
            else
                    numbersSeen[num] = true;
        }
    }
    else{
        String junk = sc.next();
    }
}
```

Initialised with false

Arrays of Counts

- To keep track of numbers we have seen, efficiently
- Counting the number of occurrences of each number a file of integers:

```
Scanner sc = new Scanner(Path.of(UIFileChooser.open("file to check")));
int[ ] numbersSeen = new int[101];
while ( sc.hasNext() ){
    if ( sc.hasNextInt() ){
        int num = sc.nextInt();
        if ( num>=0 && num<=100 ){
            numbersSeen[num] = numbersSeen[num] + 1;      //  OR     numbersSeen[num]++;
        }
    }
    else{
        sc.next();
    }
}
```

Initialised with 0