main can have a local variable and use it!

```
public static void main(String[] a)
{
   int nSevensSeen;
   nSevensSeen = 0; /*A fair count starts at 0.*/
   Scanner sc = new Scanner(System.in);
   boolean done;
   done = false; /*We are not done when we start.*/
   while( !done )
   {
      int input = sc.nextInt();
      if( input == 7 )
      {    nSevensSeen = nSevensSeen + 1; }
      if(input == 0 )
      {    done = true; }
   }
   System.out.println(“Saw “ + nSevensSeen + “ 7s”);
}
```

An object can have an instance variable (or field) and use it!

```
class House
{  public int nSevensSeen;
   public int  sum;
   public House()
   {  this.nSevensSeen = 0;
      this.sum  = 0;
      /*Our constructor method is the best place
        to code initializations like this*/ }
   public void addNumber(int theNumber )
   {   this.sum = this.sum + theNumber;
       if( theNumber == 7 )
       {  this.nSevensSeen = this.nSevensSeen + 1;
       }
   }
```

An object can have an instance variable (or field) and use it!

```
class House
{  public int nSevensSeen;
   public int sum;
   public House() {/*Body in slide above */ }
   public void addNumber(int theNumber ){ /*..*/ }
   public static void main(String[] a)
   {  House myHouse = new House( );
      Scanner sc = new Scanner(System.in);
      boolean done;
     while( !done )
     {  int input = sc.nextInt();
        myHouse.addNumber( input );
        if(input == 0 )
        {  done = true; }
     }
     System.out.println(“Saw “ +
                  myHouse.nSevensSeen + “ 7s”);
   } /*END OF main*/   } /*END of class House*/
```

An object can have a reference to an array and use it!

```
class House
{  int arrayOfInts[];  int capacity;
   int howManyInts;
   public House(int capacityParam)
   {  arrayOfInts = new int[capacityParam];
      capacity = capacityParam;
      howManyInts = 0;
   }
   public boolean addNumber(int theNumber )
   { if( this.howManyInts >= this.capacity )
     { return false;  /*avoid a crash*/ }
     this.arrayOfInts[this.howManyInts] =
            theNumber;
     this.howManyInts = this.howManyInts + 1;
     return true;  }
   public void explore( )
   {  /* show us all the numbers you got! */ }
```

An object can have a reference to an array and use it!

```
class House
{  int arrayOfInts[];  int capacity;
   int howManyInts;
   public House(int capacityParam){/* see above */}
   public boolean addNumber(int theNumber ){/*...*/}
   public void explore( )
   {  /* show us all the numbers you got! */
     for(int i=0; i < howManyInts; i++)
     {
       System.out.print(“The number in location “);
       System.out.print(        i       );
       System.out.print(      “ is “    );
       System.out.println( arrayOfInts[ i ] );
     }
}
```

main can have a local Picture reference variable and use it!

```
main(String[] a)
{  Picture p;
   p = new Picture(FileChooser.pickAPath());
   p.explore();
}
```

main can have a local array of Picture references and use it.

```
main(String[] a)
{  Scanner sc = new Scanner(System.in);
   System.out.println(“How many?”);
   int howMany = sc.nextInt();
   Picture pArray[];
   pArray = new Picture[ howMany ];
   for(int i = 0; i < howMany; i++ )
   {  pArray[ i ] =
       new Picture(FileChooser.pickAPath());
      i = i + 1;
   }
   //Imagine the rest.
}
```

Project05:

Any one Album object
can have
(1) an array of
Picture references
and
(2) use that array.

class Album is the Blueprint for every Album.
Your code in Album.java
PLANS OUT:
Storage for tracking how many Pictures are currently in it.

Storage for the Picture (reference)s

(you are learning to use the array data structure)

each Album object can have a field or instance variable locating an array of Picture references, and use it (just like today's main did!)
How?
(1) Plan it in the blueprint.

```
public class Album
{  Picture arrayOfPicts[];
   int capacity;
   int nPictsInArray;
   /* methods next */
}
```

(1) Plan it in the blueprint. (2) JUST LIKE main did, the constructor directs the computer to make the actual array with the capacity the person wants.

```
public class Album
{  Picture arrayOfPicts[];
   int capacity;
   int nPictsInArray;
   public Album(int theCapacity)
   { /* see below  */ }
}
```

The constructor directs the computer to make the actual array with the capacity the person wants.

```
public class Album
{  Picture arrayOfPicts[];
   int capacity;
   int nPictsInArray;
   public Album(int capacityParam)
   { this.arrayOfPicts =
          new Picture[capacityParam];
     capacity = capacityParam;
    nPictsInAlbum = 0;
   }
}
```

RIGHT after one Album object is constructed, BEFORE adding any Pictures, and the constructor had been called, HOW MANY PICTURES ARE IN THE ALBUM?

A) Zero
B) One
C) the value of theCapacity
D) the value of capacity
E) the length of the array

RIGHT after one Album object is constructed, BEFORE adding any Pictures, and the constructor had been called, HOW MANY PICTURES ARE IN THE ALBUM?

A) Zero
B) One
C) the value of theCapacity
D) the value of capacity
E) the length of the array

Easy to answer?!: Before any Pictures are added, there are none in the Album, so, ... the number of Pictures in the Album is 0.

What index value locates where in pictArray to copy the (ref to) the next Picture?

Mathematical fact: When array elements are located by integers beginning at 0, the LOCATION (number) for the next value to add in IS the same number as the number of values already stored in the array!

When the number of values is 0, the new value should be stored at location 0.

When the number of values is 1, the new value should be stored at location 1, etc.

It's good style to increment nPictsInAlbum = nPictsInAlbum + 1 AFTER actually storing the ref to the new Picture in pictArray[nPictsInAlbum]

You shouldn't count what's stored until after storing it is finished! Also, this discipline gives [nPictsInAlbum], the subscript expression, the simplest possible form.