ARRAYS

Outline

- Array Basics
- Creating and Accessing Arrays
- Array Details
- The Instance Variable length

Zombie Apocalypse

```
Level: 0
. . ! . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . * . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . #
Direction? s
You walked south
Zombie went east
```

How do I keep track of location of the person and the zombie?

```
int personX = 0;
int personY = 0;

int zombieX = 0;
int zombieY = 0;
```

How do I detect when the person gets eaten?

```
if ((personX == zombieX) && (personY == zombieY))
{
   System.out.println("Zombie got your braaaains!");
   gameOver = true;
}
```

Extreme Zombie Apocalypse

```
Level: 0
. . ! . . . . . . .
. . . . *. . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . .  *  . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . #
Direction? s
You walked south
Zombie went east
```

What if we need to keep track of two zombies?

```
int personX = 0;
int personY = 0;

int zombieX1 = 0;
int zombieY1 = 0;

int zombieX2 = 0;
int zombieY2 = 0;

...

if (((personX == zombieX1) && (personY == zombieY1)) ||
    ((personX == zombieX2) && (personY == zombieY2)))
{
   System.out.println("Zombie got your braaaains!");
   gameOver = true;
}
```

Super Extreme Zombie Apocalypse

```
Level: 0
. . ! . . . * . . .
. . . . * . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . * . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . #
Direction? s
You walked south
Zombie went east
```

What if we need to keep track of three zombies?

```
int personX = 0;
int personY = 0;

int zombieX1 = 0;
int zombieY1 = 0;

int zombieX2 = 0;
int zombieY2 = 0;

int zombieX3 = 0;
int zombieY3 = 0;

...

if (((personX == zombieX1) && (personY == zombieY1)) ||
    ((personX == zombieX2) && (personY == zombieY2)) ||
    ((personX == zombieX3) && (personY == zombieY3)))
{
   System.out.println("Zombie got your braaaains!");
   gameOver = true;
}
```

Zombie Apocalypse: The Rising

```
You walked south
Zombie went west
Level: 5
. * . * .
. . . * .
! . * . .
* . . . .
. . * . #
Direction?
```

What if we want to add one zombie every time the player advances a level?

No good way to do this with simple variables!

Arrays to the Rescue!

- We've already seen arrays:

```
public static void main(String [] args)
```

```
% java CostCalc bananas 12 0.21
To buy 12 bananas you will need $2.52
```

identifier	meaning	value	type
`args[0]`	1st thing on command line after Java class name	`"bananas"`	`String`
`args[1]`	2nd thing on command line	`"12"`	`String`
`args[2]`	3rd thing on command line after Java class	`"0.21"`	`String`
`args.length`	# of things on command line	`3`	`int`

Arrays: Creating Many Things

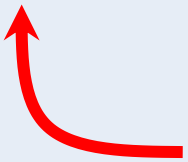
- Arrays: create many variables of same type
- Goal: Ten variables of same type
  - e.g. To hold the values 0-9

```
int a0, a1, a2, a3, a4, a5, a6, a7, a8, a9;
a0 = 0;
a1 = 1;
a2 = 2;
a3 = 3;
a4 = 4;
a5 = 5;
a6 = 6;
a7 = 7;
a8 = 8;
a9 = 9;
```

Arrays: Creating Many Things

- Arrays: create many variables of same type
- Goal: Ten variables of same type
  - e.g. To hold the values 0-9

```
int [] a = new int[10];
a[0] = 0;
a[1] = 1;
a[2] = 2;
a[3] = 3;
a[4] = 4;
a[5] = 5;
a[6] = 6;
a[7] = 7;
a[8] = 8;
a[9] = 9;
```



new keyword is used whenever we create an array

Arrays: Accessing Elements

- Arrays: we can use a variable as the index!
  - Makes code shorter, cleaner, less buggy

```
int N = 10;                                 // size of array
int [] a;                                   // declare array
a = new int[N];                             // create array

for (int i = 0; i < a.length; i++)          // initialize array elements
   a[i] = i;                                // to be 0 - 9
```

1st element of array is a[0]. We count from zero in computer science!

Arrays: Easy to Extend

- Arrays: can hold lots and lots of data
  - Same code, but now holds 100,000 integers:

```
int N = 100000;                          // size of array
int [] a;                                // declare array
a = new int[N];                          // create array

for (int i = 0; i < a.length; i++)       // initialize array elements
   a[i] = i;                             // to be 0 - 9
```

More About Array Indices

- Index of first array element is 0
- Last valid Index is **`arrayName.length - 1`**
- Array indices must be within bounds to be valid
  - When program tries to access outside bounds, run time error occurs

Gotcha – Don't Exceed Array Bounds

- The code below fails if the user enters a number like 4. Use input validation.

```
Scanner kbd = new Scanner(System.in);
int[] count = {0,0,0,0};

System.out.println("Enter ten numbers between 0 and 3.");
for (int i = 0; i < 10; i++)
{
 int num = kbd.nextInt();
 count[num]++;
}
for (int i = 0; i < count.length; i++)
   System.out.println("You entered " + count[i] + " " + i + "'s");
```

Creating and Accessing Arrays

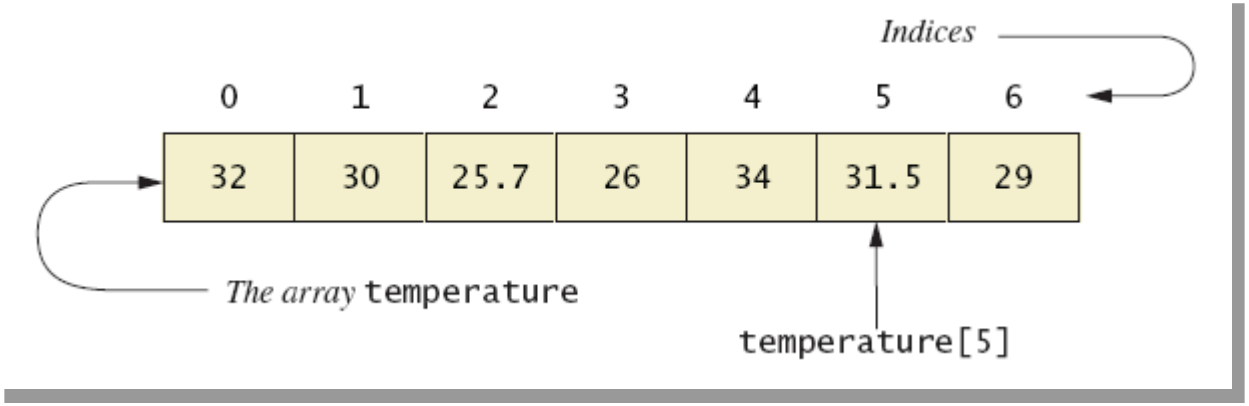
- An array is a special kind of object
- Think of as collection of variables of same type
- Creating an array with 7 variables of type double

```
double[] temperature = new double[7];
```

- To access an element use
  - The name of the array
  - An index number enclosed in braces
- Array indices begin at zero

Creating and Accessing Arrays

- A common way to visualize an array



Array Details

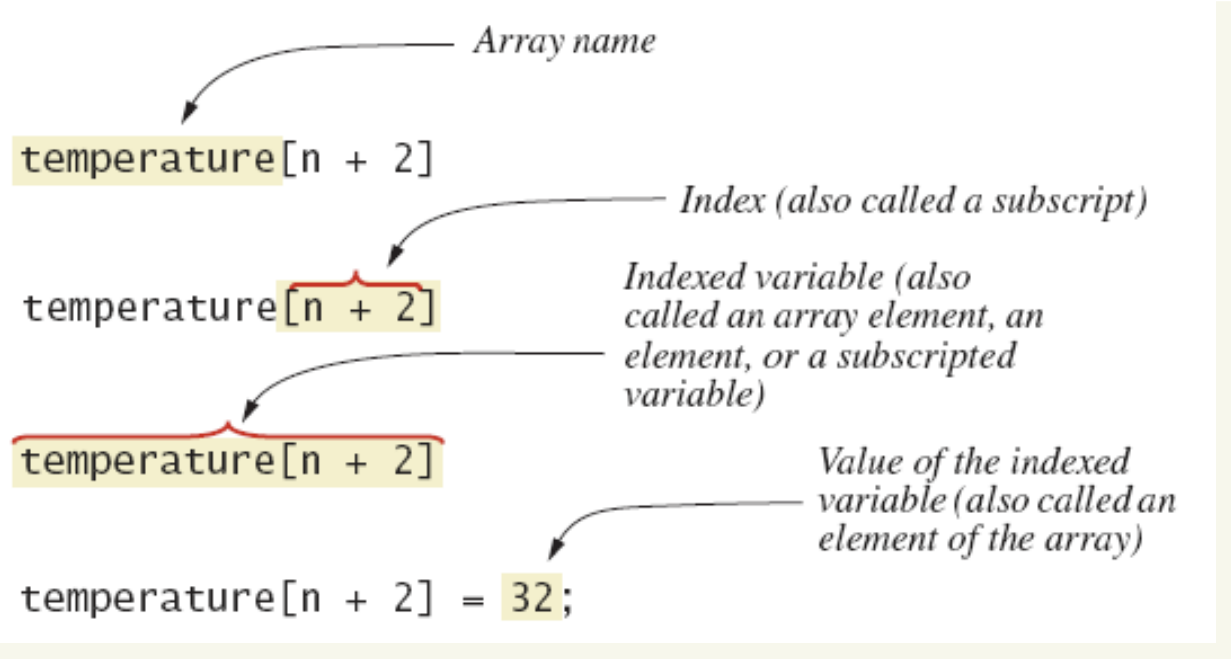
- Syntax for declaring an array with **`new`**

```
Base_Type[] Array_Name = new Base_Type[Length];
```

- The number of elements in an array is its length
- The type of the array elements is the array's base type

Array Details

- Figure 7.2 Array terminology



Square Brackets with Arrays

- With a data type when declaring an array

```
int [ ] pressure;
```

- To enclose an integer expression to declare the length of the array

```
pressure = new int [100];
```

- To name an indexed value of the array

```
pressure[3] = keyboard.nextInt();
```

Initializing Arrays

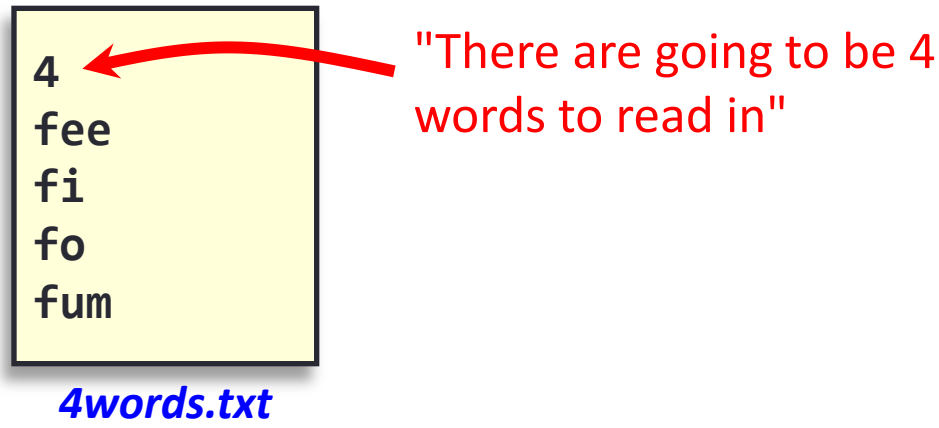
- Possible to initialize at declaration time

```
double[] reading = {3.3, 15.8, 9.7};
```

- Also may use normal assignment statements
  - One at a time
  - In a loop

```
int[] count = new int[100];
for (int i = 0; i < 100; i++)
    count[i] = 0;
```

Arrays: Loading Data from File



- Read words into array
- Print out words in reverse order

```
% java Backwards 4words.txt
fum fo fi fee
```

Arrays: Loading Data from File

```
% java PrintBackward 4words.txt
fum fo fi fee
```

```
public class Backwards
{
    public static void main(String [] args)
    {
        try
        {
            Scanner file = new Scanner(new File(args[0]));
            int num = file.nextInt();
            String [] words = new String[num];

            for (int i = 0; i < num; i++)
                words[i] = file.next();

            file.close();

            for (int i = num - 1; i >= 0; i--)
                System.out.print(words[i] + " ");

            System.out.println();
        }
        catch (FileNotFoundException e)
        {
            System.out.println("File not found.");
        }
    }
}
```

```
4
fee
fi
fo
fum
```

Super Extreme Zombie Apocalypse

What if we need to keep track of three zombies?

```
Level: 0
. . ! . . . . . .
. . . . . . . . . *
. . . . . . . . . .
. . . . . . . . . .
. . . . * . * . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . #
Direction? s
You walked south
Zombie went east
```

```
int personX = 0;
int personY = 0;
final int NUM_ZOMBIES = 3;  // constant defining # of zombies

int [] zombieX = new int[NUM_ZOMBIES]; // declare & create x-pos array
int [] zombieY = new int[NUM_ZOMBIES]; // declare & create y-pos array

// Set random initial location for each zombie (they can overlap)
for (int i = 0; i < NUM_ZOMBIES; i++)
{
   zombieX[i] = (int) (Math.random() * 10); // set i-th zombie's x-pos
   zombieY[i] = (int) (Math.random() * 10); // set i-th zombie's y-pos
}

...

int i = 0;
while ((i < zombieX.length) && (!gameOver))
{
   if ((personX == zombieX[i]) &&
       (personY == zombieY[i]))
   {
       System.out.println("Zombie got your braaaains!");
       gameOver = true;
   }
   i++;
}
```

Super Mega Extreme Zombie Apocalypse

What if we need to keep track of thirty zombies?

```
int personX = 0;
int personY = 0;
final int NUM_ZOMBIES = 30;  // constant defining # of zombies

int [] zombieX = new int[NUM_ZOMBIES]; // declare & create x-pos array
int [] zombieY = new int[NUM_ZOMBIES]; // declare & create y-pos array

// Set random initial location for each zombie (they can overlap)
for (int i = 0; i < NUM_ZOMBIES; i++)
{
   zombieX[i] = (int) (Math.random() * 10); // set i-th zombie's x-pos
   zombieY[i] = (int) (Math.random() * 10); // set i-th zombie's y-pos
}

...

int i = 0;
while ((i < zombieX.length) && (!gameOver))
{
   if ((personX == zombieX[i]) &&
       (personY == zombieY[i]))
   {
       System.out.println("Zombie got your braaaains!");
       gameOver = true;
   }
   i++;
}
```

```
Level: 0
* . ! * . . . * . .
. * . . * * . * . *
. . * . . . . . * *
* . * . . . * . . .
. . . . * . . * * *
. . * . * . . . . .
. * . . . . * . * .
. . . . . * . . . .
. . * . . . * . * .
. . . . * . . . . #
Direction? s
You walked south
Zombie went east
```

The Instance Variable `length`

- As an object an array has only one public instance variable
  - Variable **`length`**
  - Contains number of elements in the array
  - It is final, value cannot be changed

Array Assignment and Equality

- Arrays are objects
  - Assignment and equality operators behave (misbehave) as specified with other objects (e.g. String)
- Variable for the array object contains memory address of the object
  - Assignment operator = copies this address
  - Equality operator == tests whether two arrays are stored in same place in memory
- To compare the contents of two arrays, you need to:
  - See if they are the same length
  - Use a loop to compare the contents element by element
- What about the equals method?

Summary

- Array Basics
- Creating and Accessing Arrays
- Array Details
- The Instance Variable length