## Initialization

- initialization and declaration can be combined in a single statement

```
int count;    // Declare a variable named count.
count = 0;    // Give count its initial value.
```

```
int count = 0;  // Declare count and give it an initial value.
```

- multiple variables of the same type can be declared (and initialized) in a single statement

```
char firstInitial = 'D', secondInitial = 'E';

int x, y = 1;   // OK, but only y has been initialized!
```

- for arrays, there are two initialization steps
  - make the array only

```
String[] nameList = new String[100];
```

  - make the array and initialize each slot (initializer list syntax)

```
int[] smallPrimes = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 };
```



## Named Constants

- the idea is to give a meaningful name to what would otherwise be just a literal value

```
int key = getKey();

if ( key == 2 ) { … }
if ( key == LEFT ) { … }
```

  - the second form makes it easier to read (and write) the program
  - the second form makes it easier to change the value
- use named constants when
  - the value associated with a meaning is arbitrary
    - e.g. using 2 to represent left
  - a value which looks arbitrary has a meaning (and might be changed in the future)
    - e.g. the dimensions of the grid, the width and height of a square, the number of mines



```
public class ConstantsDemo {

    public static void main(String[] args) {
        final int NUMDICE = 5, NUMSIDES = 6;

        int[] dice = new int[NUMDICE];

        // roll dice
        for (int i = 0; i < dice.length; i++) {
            dice[i] = (int) (Math.random() * NUMSIDES) + 1;
        }

        // print dice, all on one line
        for (int i = 0; i < dice.length; i++) {
            System.out.pr
        }
        System.out.println(
    }
}
```

```
public class ConstantsDemo2 {

    public static final int NUMDICE = 5, NUMSIDES = 6;

    public static void main(String[] args) {
        int[] dice = new int[NUMDICE];

        // roll dice
        for (int i = 0; i < dice.length; i++) {
            dice[i] = (int) (Math.random() * NUMSIDES) + 1;
        }

        // print dice, all on one line
        for (int i = 0; i < dice.length; i++) {
            System.out.print(dice[i] + " ");
        }
```



## Named Constants

- constants can be local variables inside main, another subroutine, or a block (e.g. a loop body)
  - prefer local constants when the scope is limited
- more commonly constants are global
  - prefer global when constants are referenced in multiple subroutines within a class or in multiple classes
  - declared outside subroutines (but inside a class)
  - `static`
- `final` is what makes a variable a constant
  - the value can't be changed once initialized
  - like putting a locked glass lid over the box – can see the value but can't replace it with something new
- convention is that constant names are ALL_CAPS (with _ to separate words)

# Named Constants

- local constants and `static` global constants declared in the same class as they are used can be referenced with just the name
  - e.g. `WIDTH` or `NUM_MINES`
- `static` global constants declared in a different class must be referenced using the class name where they are declared
  - e.g. `Color.RED`