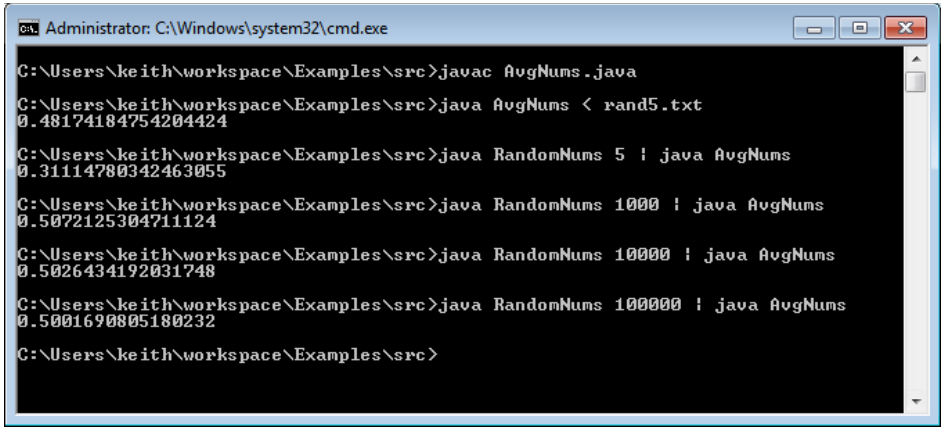
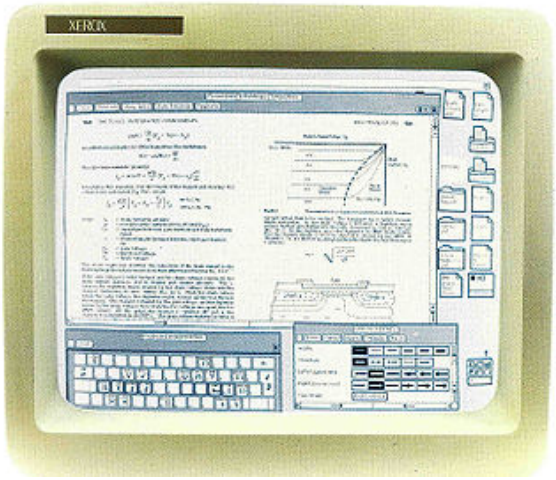
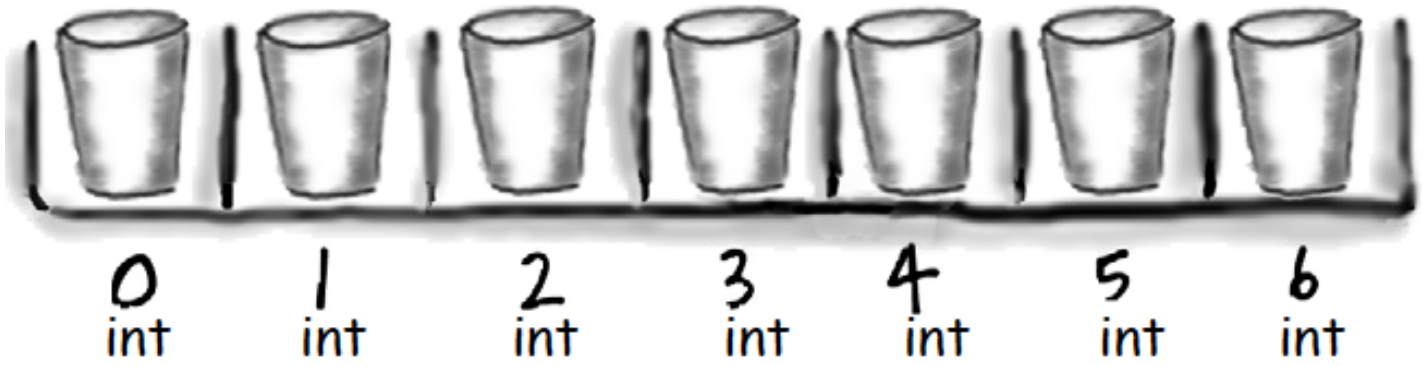
Command line, standard input, and arrays





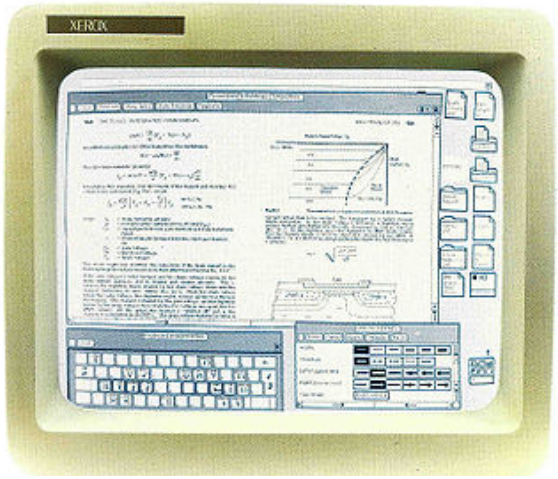


Overview

- Learning to use the command line
- New ways to get input into your programs:
  - Read information from a file
  - Read information from another program
- New way to store things
  - Arrays: store multiple things under one name
  - e.g. args[0], args[1], args[2]

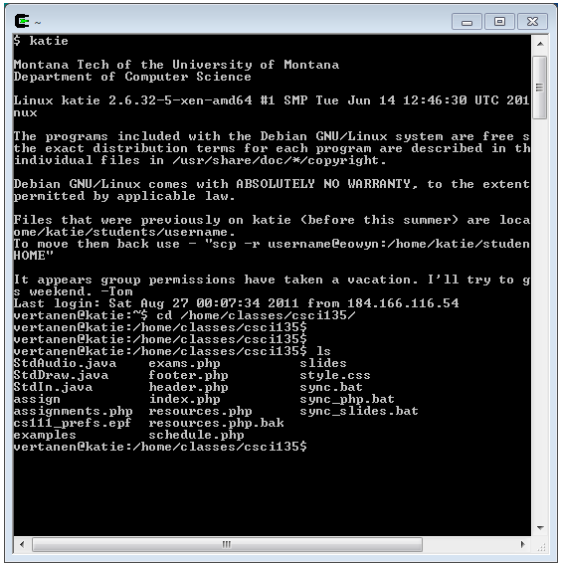
Interfacing with your computer

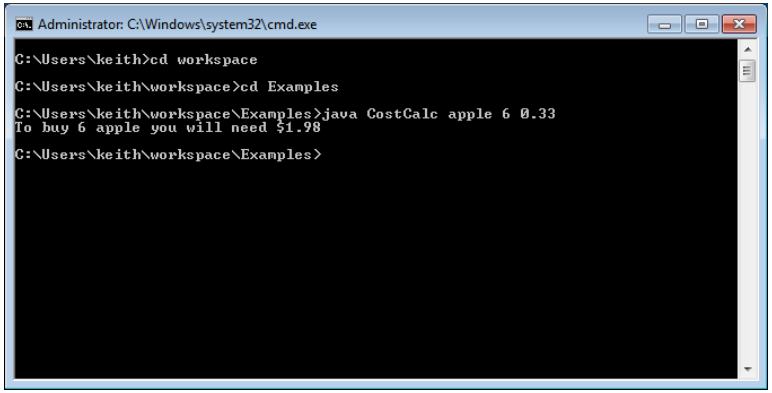
- GUI (graphical user interfaces)
  - Today: predominant interaction method
  - Windows, buttons, mouse
  - Advantages
    - Easier for novices
    - No commands to remember
    - Rich input and output capabilities



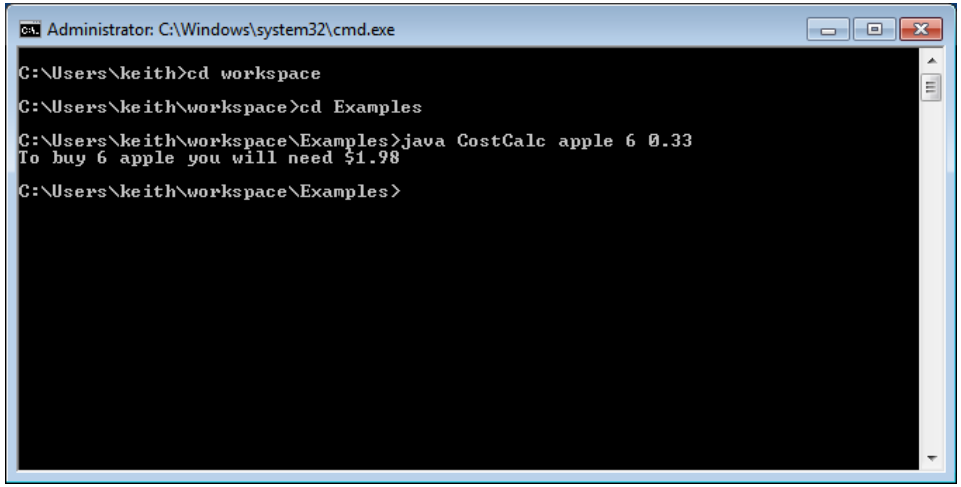
Interfacing with your computer

- Command line interface (CLI)
  - Originally the only option
  - Input by typing commands
  - Advantages:
    - Can be faster for experts than a GUI
    - Easier to automate tasks
    - Easier to hook programs together





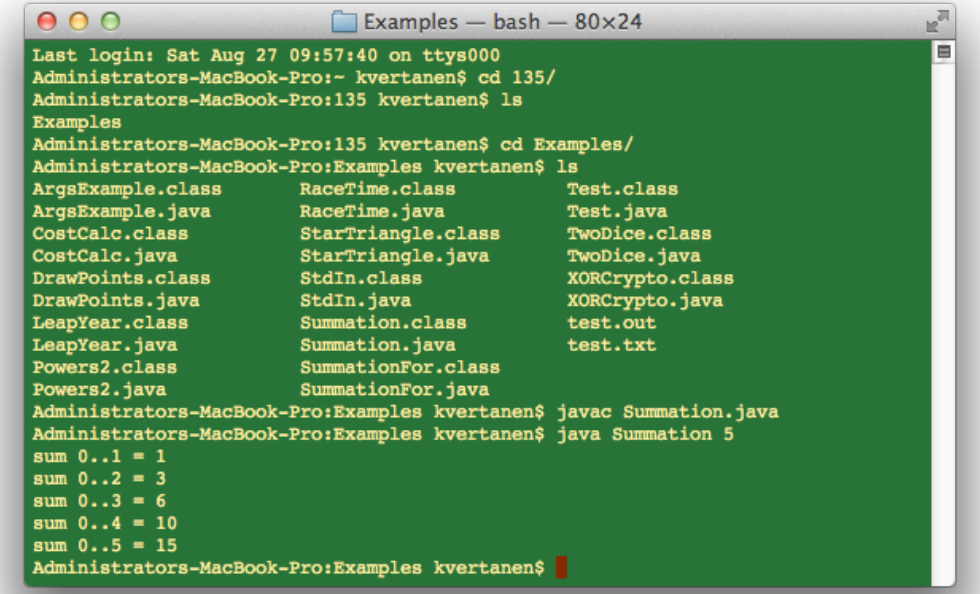
Starting a command shell



Windows

Start → type "cmd"

All Programs → Accessories → Command Prompt

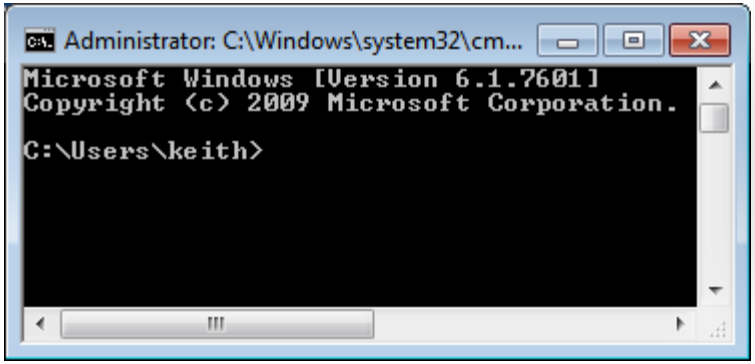


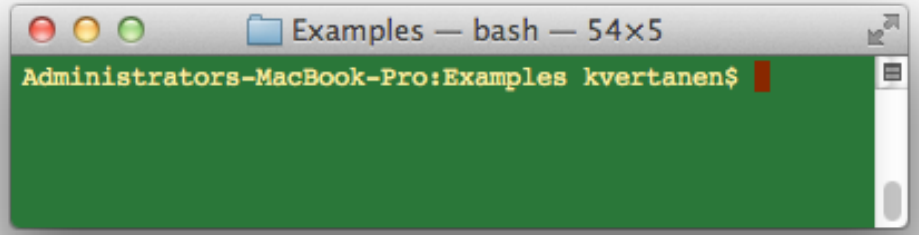
Mac

Spotlight → type "terminal"

Go → Applications → Utilities → Terminal

Getting around the command line





Action	Windows	Mac OS / Unix
Move into a folder	`cd myfolder`	`cd myfolder`
Move into parent folder	`cd ..`	`cd ..`
Move into a folder, absolute folder	`cd \Users\keith`	`cd /Users/keith`
List files in current folder	`dir`	`ls`
Compile program in current folder	`javac Prog.java`	`javac Prog.java`
Run a compiled program	`java Prog`	`java Prog`
See what is in a text file	`type Prog.java`	`more Prog.java`
Auto-complete filenames	`<tab key>`	`<tab key>`
Last command	`<up arrow>`	`<up arrow>`

Input via command line

- Input via args[] array
  - Tedious to enter lots of input
  - Impossible to have interactive user input
  - e.g. What we need for `NumberHunt.java`

```
% java NumberHunt
Guess a number between 1-100? 50
Ice cold.
Guess a number between 1-100? 20
Getting warmer.
Guess a number between 1-100? 10
Hot.
Guess a number between 1-100? 5
Getting warmer.
Guess a number between 1-100? 15
Hot.
Guess a number between 1-100? 12
You nailed it!
It took you 6 guesses.
```

Standard input class

- Allows input from user or from a file
- Download `StdIn.java`
  - Place in same directory as your program

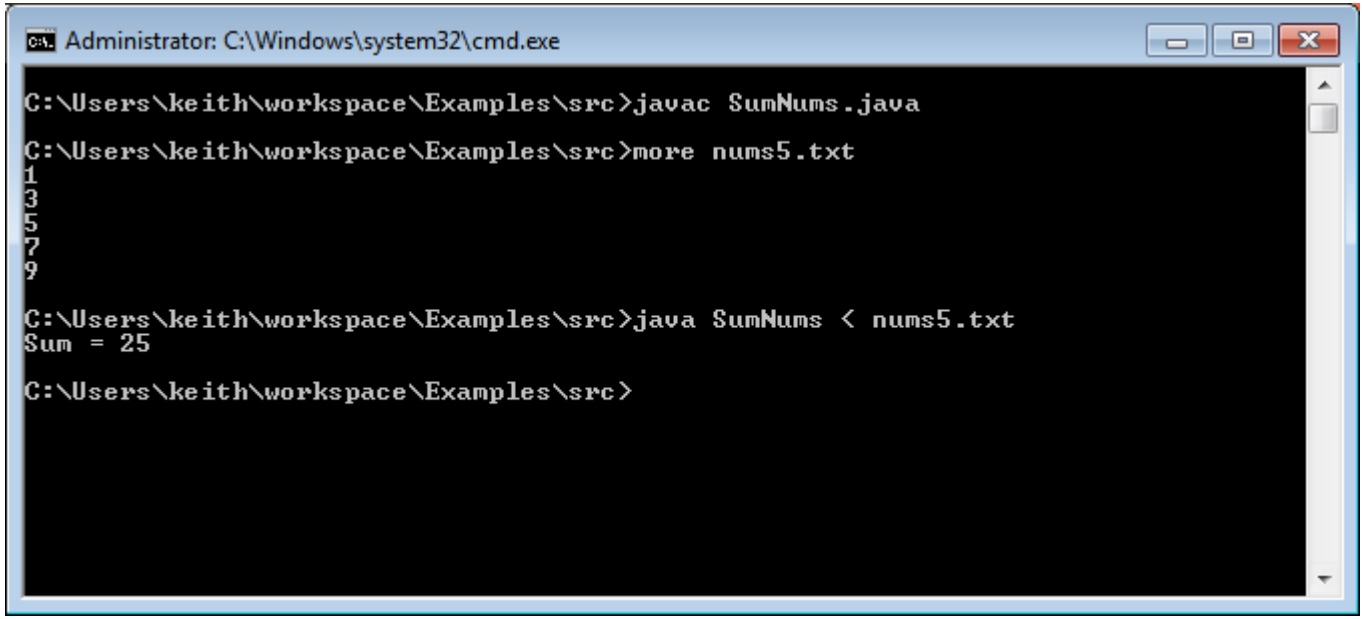
```
public class AddTwo
{
   public static void main(String [] args)
   {
      System.out.print("Enter first integer: ");
      int num1 = StdIn.readInt();

      System.out.print("Enter second integer: ");
      int num2 = StdIn.readInt();

      int sum = num1 + num2;
      System.out.println("Sum = " + sum);
   }
}
```

Standard input class

- Reading from a file via redirection
  - Need to do from command line
- Example: sum all integers in a file

```
public class SumNums
{
   public static void main(String [] args)
   {
      int sum = 0;
      while (!StdIn.isEmpty())
      {
         sum += StdIn.readInt();
      }
      System.out.println("Sum = " + sum);
   }
}
```

Reading from a file



StdIn.java

```
public class StdIn
```

`boolean`	*`isEmpty`*`()`	`true` if no more values, `false` otherwise
`int`	*`readInt`*`()`	read the next `int`
`double`	*`readDouble`*`()`	read the next `double`
`long`	*`readLong`*`()`	read the next `long`
`boolean`	*`readBoolean`*`()`	read the next `boolean`
`char`	*`readChar`*`()`	read the next `char`
`String`	*`readString`*`()`	read the next `String`
`String`	*`readLine`*`()`	read the rest of the line (until a carriage return)
`String`	*`readAll`*`()`	read the rest of the text

```
this is an example text file
1.23 3.45
10 20
the
end
```

Combining programs

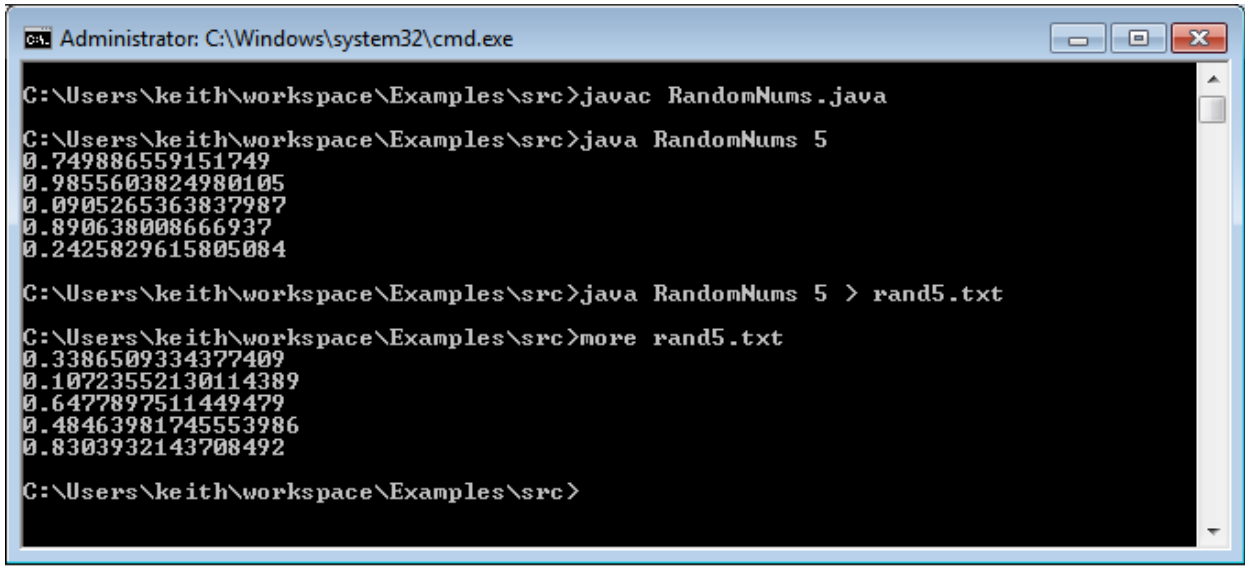
- Output can also be redirected
  - To a file (for later review) via redirection
  - Directly to another program via piping
- Example:
  - First program generates random numbers
  - Second program averages the numbers

Combining programs

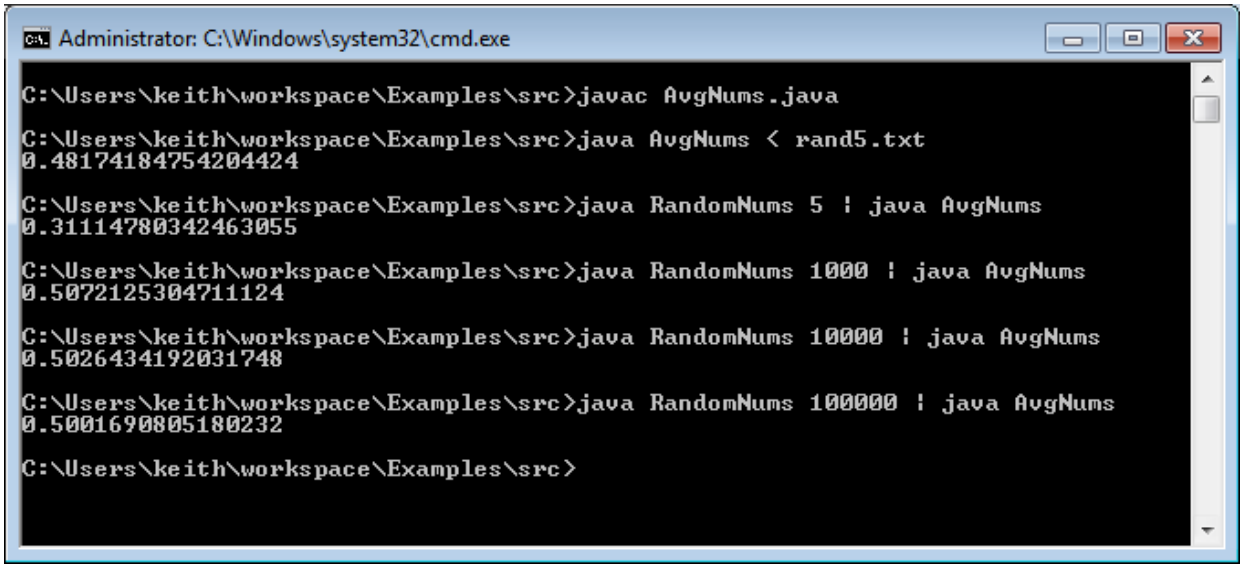
```
public class RandomNums
{
   public static void main(String [] args)
   {
      int num = Integer.parseInt(args[0]);
      for (int i = 0; i < num; i++)
         System.out.println(Math.random());
   }
}
```

```
public class AvgNums
{
   public static void main(String [] args)
   {
      double sum   = 0.0;
      long   count = 0;
      while (!StdIn.isEmpty())
      {
         sum += StdIn.readDouble();
         count++;
      }
      System.out.println(sum / count);
   }
}
```

Averaging random numbers



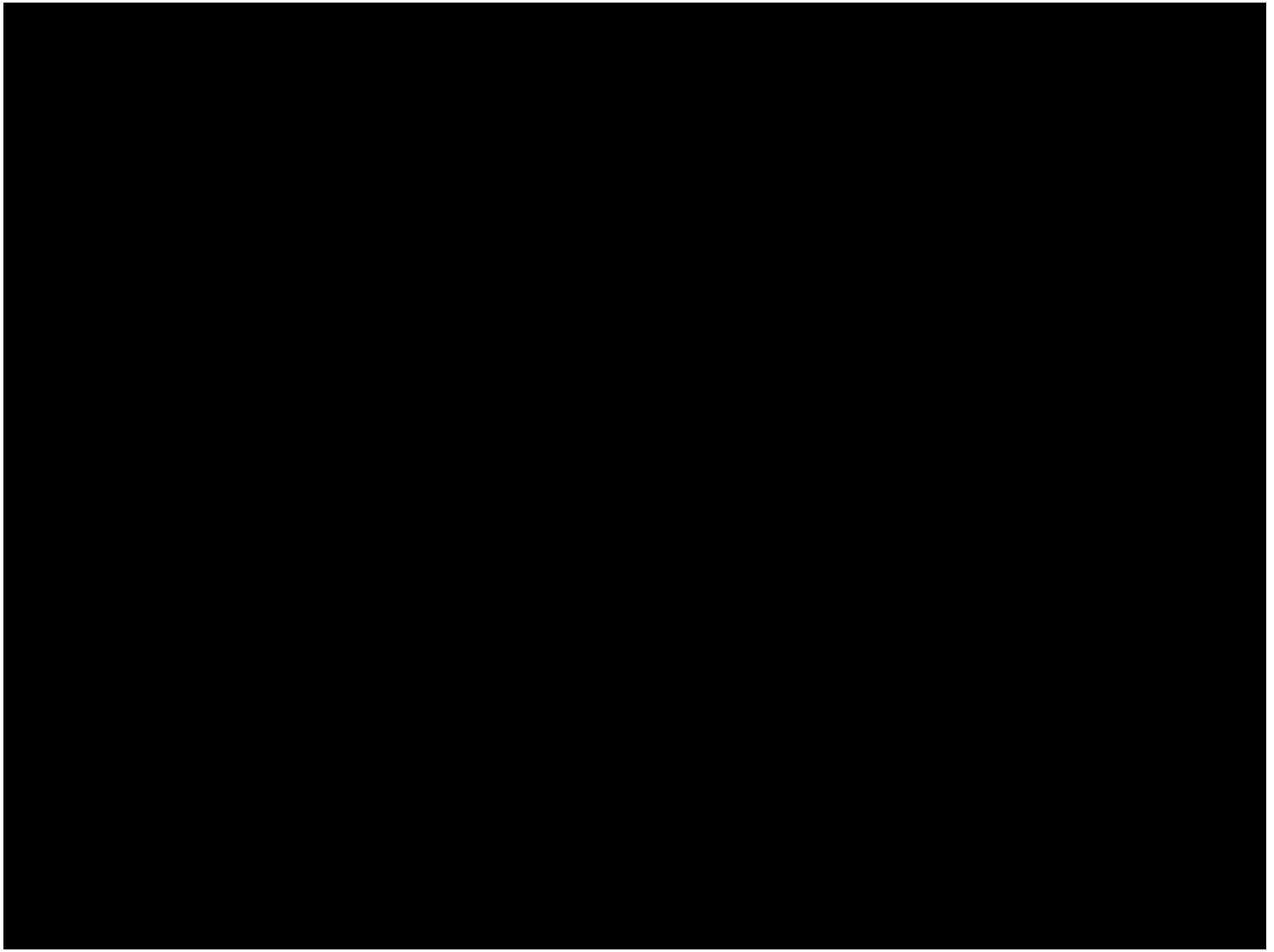
Redirecting program output to a file using ">" followed by the output filename.



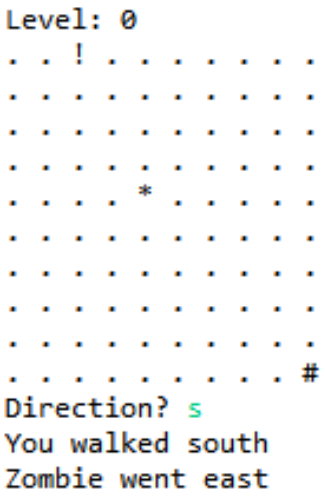
Reading input from file using "<" followed by the filename.

Directly piping output from one program to another using pipe "|"

Zombie Apocalypse



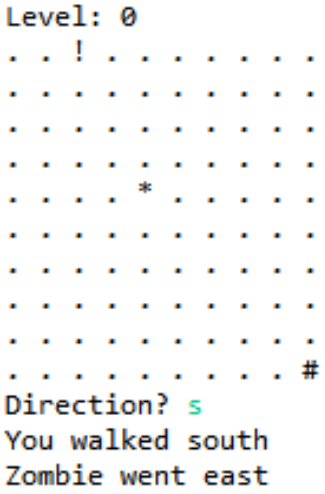
Zombie Apocalypse



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

How do I detect when the person gets eaten?

Zombie Apocalypse



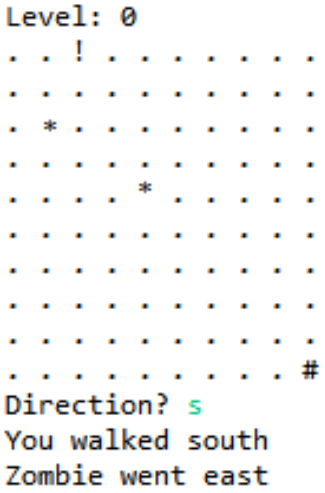
How do I keep track of location of the person and the zombie?

```
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;
}
```

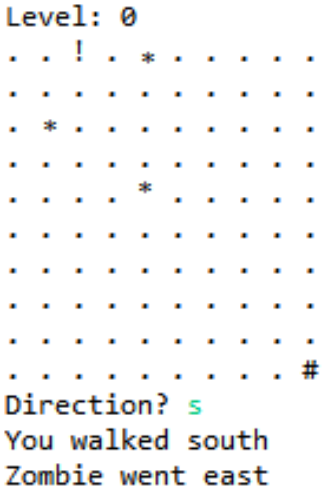
Extreme Zombie Apocalypse



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



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 holding 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 holding 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;
```

Arrays: accessing elements

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

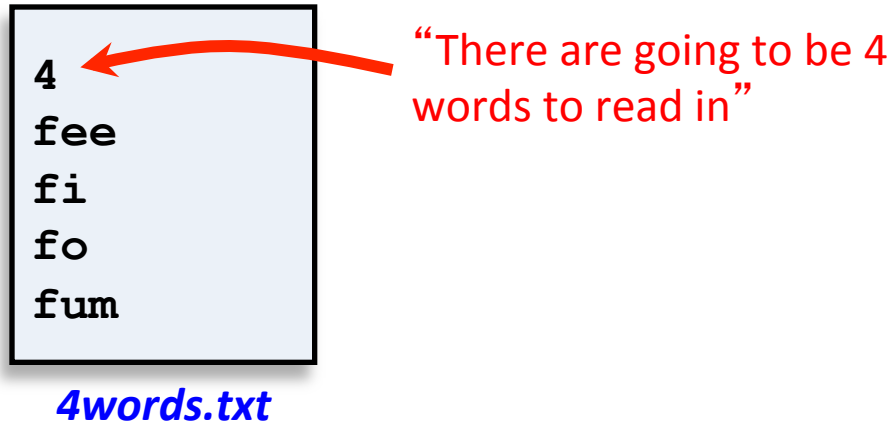
```
int N = 10;                                  // size of array
int [] a;                                    // declare array
a = new int[N];                              // create array

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

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
```

Arrays: loading data from file



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

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

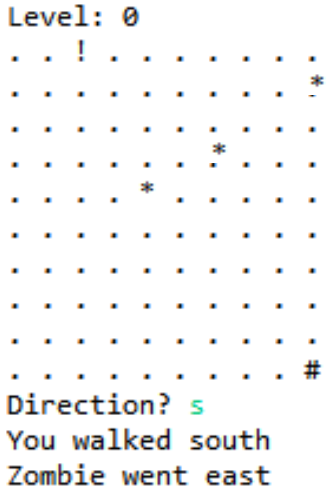
Arrays: loading data from file

```
4
fee
fi
fo
fum
```

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

```
public class Backwards
{
   public static void main(String [] args)
   {
      int num = StdIn.readInt();
      String [] words = new String[num];
      for (int i = 0; i < num; i++)
         words[i] = StdIn.readString();
      for (int i = num - 1; i >= 0; i--)
         System.out.print(words[i] + " ");
      System.out.println();
   }
}
```

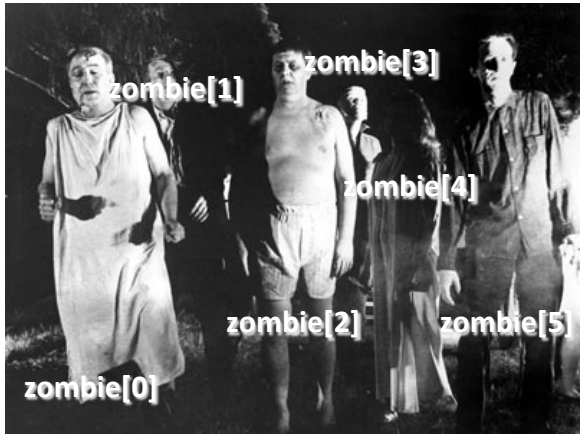
Super Extreme Zombie Apocalypse



What if we need to keep track of three zombies?

```
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++;
}
```

Summary



- Command line
  - Redirect output to a file
  - Redirect input from a file
  - Pipe output between programs
- Standard input
  - Easy way to read from user or file
- Arrays
  - Allow easy storage of similar data
  - Crucial for developing more advanced programs