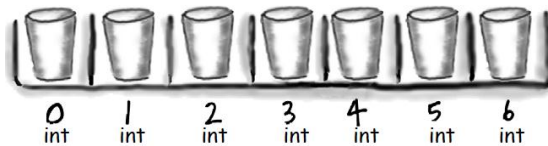
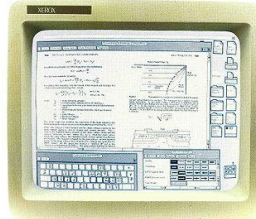
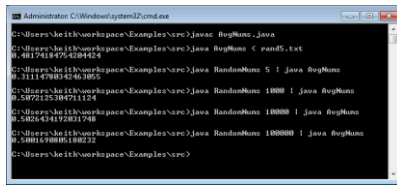


Command line, standard input, and arrays



Fundamentals of Computer Science • Keith Vertanen • Copyright © 2013

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



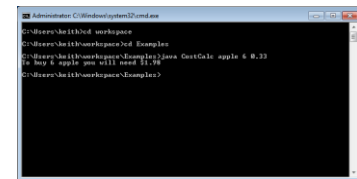
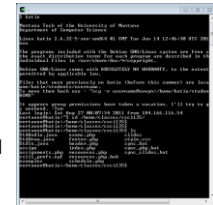
3

Overview

- Learning to use the **command line**
- **New ways to get input** into your programs:
 - Read information **from user**
 - 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

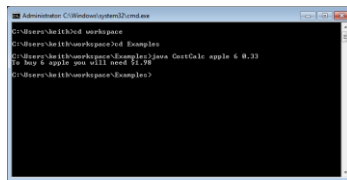
- **Command line interface (CLI)**
 - Originally the only option
 - Input by **typing commands**
 - Advantages:
 - Can be **faster for experts** than a GUI
 - Easier to **automate** tasks
 - Easier to **hook programs together**



2

4

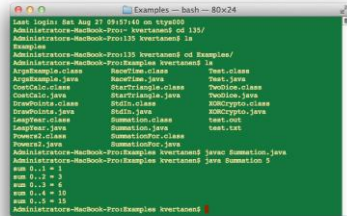
Starting a command shell



Windows

Start → type "cmd"

All Programs → Accessories → Command Prompt

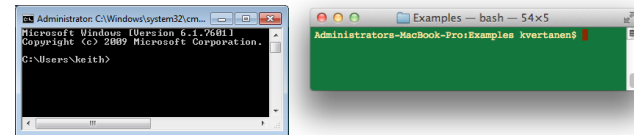


Mac

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 a number hunting game

```
% 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
 - Refresh Eclipse project to make it show up

```
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);
    }
}
```

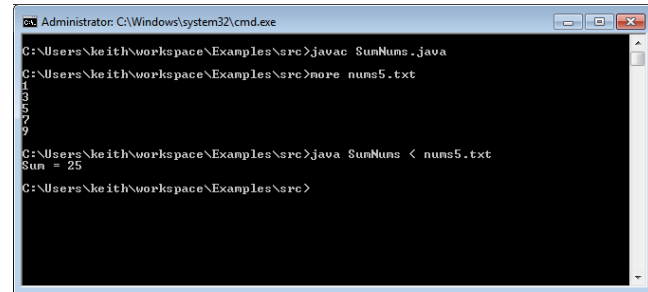
Standard input class

- Reading from a file via **redirection**
 - Need to do from command line
 - Can't redirect file (easily) inside Eclipse
- **Goal: Sum all integers in a file**
 - Keep reading numbers until End Of File (EOF)
 - EOF can be sent by hitting ctrl-z or ctrl-d in Eclipse

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

9

Reading from a file



```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\keith\workspace\Examples\src>javac SumNuns.java
C:\Users\keith\workspace\Examples\src>more nuns5.txt
1
3
5
7
9
C:\Users\keith\workspace\Examples\src>java SumNuns < nuns5.txt
Sum = 25
C:\Users\keith\workspace\Examples\src>
```

10

StdIn.java

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

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

11

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

12

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;
```

21

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;
```

22

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!

23

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
```

24

