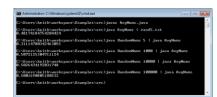
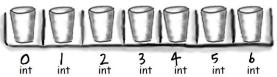
# Command line, standard input, and arrays







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





### 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]

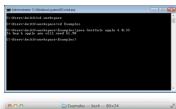
# Interfacing with your computer

- 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



,

### Starting a command shell



#### Windows

Start → type "cmd"

All Programs → Accessories → Command Prompt



#### Mac

Spotlight → type "terminal"

 $Go \rightarrow Applications \rightarrow Utilities \rightarrow Terminal$ 

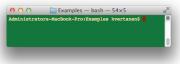
### Input via command line

- 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.
```

# 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>	<tab key=""></tab>
Last command	<up arrow=""></up>	<up arrow=""></up>

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

### 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
                          read rest of line (until carriage return)
String
           readLine()
String
          readALL()
                          read the rest of the text
```

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

# Reading from a file

```
Administrator C:\Windows\system32\cmd.exe

C:\Users\keith\workspace\Examples\src\javac SunNums.java

C:\Users\keith\workspace\Examples\src\nore nums5.txt

1
3
5
7
9

C:\Users\keith\workspace\Examples\src\java SunNums < nums5.txt

Sun = 25
C:\Users\keith\workspace\Examples\src\java SunNums < nums5.txt
```

10

### 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);
    }
}</pre>
```

# Averaging random numbers

```
Redirecting program
Cr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.java
Cr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 5

Cr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 5

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 5

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 5

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 5

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 5

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 5

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 5

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 5

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 5

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 6

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 1000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 1000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 1000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 10000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 1000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 10000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 10000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 10000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 10000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 100000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 100000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 100000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 100000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 100000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 100000

Gr. Usera: No. 16 hower kepace \Samples \sec_java & Randon\unus.s. 1000000
```

;

15

### Enter the zombies...



# **Zombie Apocalypse**

14

# Extreme Zombie Apocalypse

```
Level: 0
                        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;
. . . . . . . . . #
Direction? s
You walked south
                        if (((personX == zombieX1) && (personY == zombieY1)) ||
                            ((personX == zombieX2) && (personY == zombieY2)))
Zombie went east
                          System.out.println("Zombie got your braaaains!");
                           gameOver = true;
```

# Super Extreme Zombie Apocalypse

```
Level: 0
                        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:
Direction? s
                        int zombieY3 = 0;
You walked south
Zombie went east
                        if (((personX == zombieX1) && (personY == zombieY1)) ||
                            ((personX == zombieX2) && (personY == zombieY2)) ||
                            ((personX == zombieX3) && (personY == zombieY3)))
                           System.out.println("Zombie got your braaaains!");
                           gameOver = true;
```

18

# Zombie Apocalypse: The Rising

17

### 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]	2 <sup>nd</sup> thing on command line	"12"	String
args[2]	3 <sup>rd</sup> 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: accessing elements

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



# 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

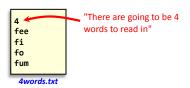
```
int [] a = new int[10];
a[0] = 0;
a[1] = 1;
a[2] = 2;
a[3] = 3;
a[4] = 4;
a[5] = 5;
a[6] = 6;
a[7] = 7;
a[8] = 8;
a[9] = 9;
```

24

### Arrays: easy to extend

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

# Arrays: loading data from file



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

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

25

# Super Extreme Zombie Apocalypse

```
What if we need to keep track of three zombies?
Level: 0
::!::::::
                 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)
Direction? s
                  for (int i = 0; i < NUM ZOMBIES; i++)</pre>
You walked south
Zombie went east
                     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))</pre>
                     if ((personX == zombieX[i]) &&
                         (personY == zombieY[i]))
                         System.out.println("Zombie got your braaaains!");
                         gameOver = true;
                                                                                       27
```

# Arrays: loading data from file

```
% java PrintBackward < 4words.txt
fum fo fi fee
fum
for 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();
    }
}
```

-

### 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