

CS126 Precept 6

Meeting 3

Randy Wang

HW 0 Feedback: Indentation

- Stuff at the “same level” not indented
- Body indented:
 - Function body (including variable declarations)
 - Loop body
 - If body else body
- Restore to previous indentation level when coming out one of these “bodies”

HW0 feedback: When to initialize variables to zero?

- Simple rule: by the time you read a variable, the variable must have been assigned a value

```
double x;  
double sum=0;  
for (j = 0; j < 5; j++) {  
    x = Math.random();  
    sum += x;  
}
```

no need to do x=0 at the top.
But sum=0 is a must.

HW 0 Feedback: Input Methods

- Three input methods so far:
 - Command line arguments
 - User input (standard input) after program starts
 - Input files
- All of them allow program to do different tasks based on different inputs

HW 0 Feedback: Input Methods

- Command line arguments:
 - If small amount of arguments are obviously known before program starts
- User input (standard input)
 - If the user could benefit from more prompting
 - If input may need to vary based on interaction with the program
- Input files
 - If there's a large amount of input data

Redirecting Standard Input/Output

- **java Prog**
 - Reads from keyboard, outputs to terminal window
- **java Prog < inputfile.txt**
 - Reads from file, outputs to terminal window
- **java Prog > outputfile.txt**
 - Reads from keyboard, outputs to file
- **java Prog < inputfile.txt > outputfile.txt**
 - Reads from file, outputs to file

Pipes connecting programs

- `java Prog1 | java Prog2`
 - Prog1 reads from keyboard
 - Output of Prog1 becomes input of Prog2
 - Output of Prog2 goes to terminal
- `java Prog | java Prog`
- `java Prog | java Prog | java Prog`

Mixing Standard I/O Redirection and Pipes

- `java Prog1 < inputfile.txt | java Prog2`
 - Prog1 reads from file
 - Output of Prog1 becomes input of Prog2
 - Output of Prog2 goes to terminal
- `java Prog1 < inputfile.txt | java Prog2 > outputfile.txt`
 - Like the above, but final output goes to file

What's all this good for?

- In the old days: hard-code input/output devices into programs
- Hard to program
- and very hard to port to different input/output devices

What's all this good for?

- Along came Unix (early 1970s)
- First OS to have complete features of standard I/O redirection and pipes
- Standard I/O redirection
 - Write program once
 - Same program can be made to work for different input/output devices at run time

What's all this good for?

- Pipes
 - Write small programs that specialize in very simple tasks
 - Connect lots of smaller programs to make bigger programs
 - Makes bigger programs easier to write
 - Earliest and best success story of programming with components
- Standard I/O redirection and pipes: big part of Unix success

Turtle.java

- `Turtle.create(512,512);`
- `Turtle.spot("earth.gif");`
- `Turtle.spot(20);`
- `Turtle.fly(x,y);`
- `Turtle.grunt("laser.wav");`
- `import java.awt.Color;`
- `Turtle.clear(Color.black);`
- `Turtle.setColor(Color.red);`