CS 113: Computer Science I

Week 2, Lecture 1

Announcements

Today's lab:

- ssh practice
- Finish Lab #1
- Start Lab #2

Homework #1:

- Releases tomorrow (Wednesday)
- Due next Tuesday at midnight on gradescope

I will post slides tonight, but they won't have all the in-lecture examples immediately.

Intros

Speed of lecture / posting slides

Today’s Topics

Review: Directory navigation, variables, printing, math operators, and String[] args

Reading in data

Logical operators

Booleans and conditionals

Navigating a Directory

`ls`: list files in current directory.

`cd`: change current directory.

`pwd`: print path to the current directory.

`mkdir`: make a directory.

Purpose of Directories

- Organization: group related files together
- Name isolation: different files can share the same name as long as they live in separate paths or folders
- Security: apply permission settings to control who can view, edit, or delete files inside a specific directory (ex: /home/yourusername directories on lab machines)
- Efficiency: help the OS quickly locate and retrieve stored data without reading the entire disk

Command Line Prompt

All of the text that's on the left of where you are typing commands is called the prompt.

So here: `kshiptoski@comet:~$`

Variables

Have a data type, name, and (possibly) a value.

```
int x = 7;
```

Printing Variables

```
class Hello {
  public static void main(String[] args) {
    int a = 1;
    double b = 2.0;
    String blah = "thisisastring";

    System.out.println(a);
    System.out.println(b);
    System.out.println(blah);
  }
}
```

```
1
2.0
thisisastring
```

Printing Variables

```
class Hello {
  public static void main(String[] args) {
    int a = 1;
    double b = 2.0;
    String blah = "thisisastring";

    System.out.println("a is " + a + ", b is " + b + ", and blah is " + blah);
  }
}
```

```
a is 1, b is 2.0, and blah is thisisastring
```

Rules for Naming Variables

Case sensitive.

Can’t:

- Start with a number.
- Contain special characters (*, +, /, etc).
- No spaces.
- No special words (String, int, main, etc).

Converting Types (Numbers)

Double to integer:

- (int) 3.14;
- int a = (int) 3.14; // Store the converted double in a var

Storing an integer as a double:

- double b = 6;

Converting Types (Strings & Numbers)

Integer to String

- int a = 23;
- String numMajors = String.valueOf(a);

String to integer

- int x = Integer. parseInt(“40”);

String to double

- double a = Double.parseDouble(“40.11”);

Basic Operators

Operators: +, -, /, *, %...

Expressions:

- Example: 55 + c
- Operator is **+**
- Operands: **55**, **c**

Modulo Operator

Operator: %

x % y ➜ returns remainder of x / y

Example: 48 % 2 ➜ 0

String Operators

Concatenation: combining strings together.

Concatenation operator: +

```
System.out.println(“Hi there ” + name);
```

Math Utilities

Math.round(40.11);

Math.sqrt(9);

Math.random();

Exercise

Expression	Value	Data Type
-4		
3.76		
“42.64”		
10 + 3.3		
9 - 5 * 1		
“hot” + “dog”		

```java
class Practice {
  public static void main(String[] args) {
    // What if the user doesn't pass an argument?
    String firstArg = args[0];

    double miles = Double.parseDouble(firstArg);
    // double miles = 72.0;
    double km = miles * 1.6;
    System.out.println(miles + " miles is approximately " + km + " km");
  }
}
```

Reading in Data

What if the user doesn't know they have to pass an argument?

- Have to check to prevent errors.
- User would have to know to pass in the number of miles.

It would make more sense for our program to prompt the user for the number of miles they want converted to km.

Scanner Class

A nice way to read in data. Allows us to **prompt** the user for input.

```
Scanner input = new Scanner(System.in);
```

What is `System.in`?

What type is `input`?

Using Scanner

Read in a line from the user: nextLine();

- Data type: String

More on these later:

- Read in a double: nextDouble();
- Read in an integer: nextInt();

Let’s refactor our miles to kilometer program to use Scanner.

Example: Miles to Kilometers

Example: Echo Program

Booleans & Conditionals

New data type.

Two possible values:

- true
- false

Example: `bool isWet = true;`

Allows us to create **conditional expressions**.

Expressions

Expression:

```
bool isWet = true;
```

Conditional expression:

```
if isWet == true
```

Conditional Expressions & Relational Operators

Conditional expressions produce **true** or **false** values.

Relational operators:

- >
- >=
- <
- <=
- ==
- !=

Watch out for == vs =.

x = 4;

x == 4

Exercise: Relational Expressions

```
int temp = 68;
double val = 10.5;
boolean raining = true;
```

Expression	Value	Data Type
`temp > 80`		
`val != 5.6`		
`val >= 10.1`		
`raining == true`		
`raining`		
`raining == false`		

Logical Operators

Logical operators allow us to **combine Boolean expressions**.

Logical operators:

Operator	Meaning	Example
&&	and	isRaining && isCold
||	or	x == 4 || x ==5
!	not	!isRaining

Rules of Logical Operators

X && Y is true when **both** X and Y are **true**.

X || Y is true when **either** X or Y is **true**.

!X is true when X is **false**.

!X is false when X is **true**.

Exercise: Logical Expressions

```
boolean isHappy = true;
boolean knowIt = false;
int temp = 40;
```

Expression	Value	Type
isHappy && knowIt		
isHappy		
isHappy || temp > 80		
isHappy || knowIt		
!knowIt		
isHappy && (temp < 80 || !knowit)		

Decision Making: if / else

Branching decision-making based on values of Booleans expressions.

Execute different portions of the code in different circumstances.

```
boolean isHappy = true;
boolean knowIt = false;

if (isHappy && knowIt) {
    System.out.println(“Clap your hands!”);
} else {
    System.out.println(“Sit quietly.”);
}
```

Exercise: IsEven

TODO: future: nextint nextline trap