Lecture 02: Elementary programming
Schedule

• Housekeeping
• Chapter 1
• Lecture content
  • Algorithms
  • Variables and data types
  • Mathematical operations
• Read input from console
Housekeeping

- Everyone able to access NYU classes?
- Office hours
- Eclipse problems?
- Everyone get the book?
- Homework questions?
Go over from chapter 1

• What is a class?
• What is a method?
• symbols we’ve seen
• Good style - section 1.9
• Errors
• Describes the steps of how to solve a problem in code.

• Like a recipe, you define the inputs and outputs and steps to take to get there, in order.

• Can write this as pseudocode, right in your editor as comments
Start a new eclipse project
public class TipCalculator {

    public static void main(String[] args) {
        // Set up variables

        // Calculate tip total

        // Add tip total to food total

        // Print out result
    }
}
Variables

- Represent values that may change in the program
- Variables need to be declared before they are used. You tell the computer it's name and what type of data it will hold.
- Variable names should be descriptive of the data it's holding
Variables

• Syntax:   datatype variableName;
  Example:   int characterCount;

• Assignment using “=”
  Example:   y = 1;

• You can initialize a value when you declare it all in one go:
  Example:   int y = 1;

• Note: lowercase first word, uppercase the rest.
Variables

- *Primitive* datatypes in Java are:
  - `int`
  - `double` (floating-point number, has a decimal)
  - `char`
  - `boolean`
Constants

• Will not change in your program

• Permanent data.

• Example: final double PI = 3.14159;

• Constants are declared “final”, and they are written in all caps
Let’s talk a minute about naming

- **Identifiers** are names for things. Classes, methods, variables all need identifiers.

- Sequence of letters, digits, _ and $.

- Can’t start with a digit.

- Can’t be a reserved word. (Eclipse will probably complain if you try.)
Identifiers - which are valid?

• _randomNumber

• 42IsTheAnswer

• $2

• x&y
Identifiers

• case sensitive - so total, Total, TOTAL, ToTaL are all different

• try to avoid abbreviations. This is a good stylistic choice to follow

• don’t actually use $. It’s weird and kinda used just by generated code.
Identifiers

- Start with lowercase for variables and methods (e.g. computeTotal, radius)

- Start with Capital for classes (e.g. Ball, Canvas)

- Capitalize every letter and use underscore for constants (e.g. MAX_RESOLUTION, PI)
# Numeric data types

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>Storage Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>$-2^7$ to $2^7 - 1$ ($-128$ to $127$)</td>
<td>8-bit signed</td>
</tr>
<tr>
<td>short</td>
<td>$-2^{15}$ to $2^{15} - 1$ ($-32768$ to $32767$)</td>
<td>16-bit signed</td>
</tr>
<tr>
<td>int</td>
<td>$-2^{31}$ to $2^{31} - 1$ ($-2147483648$ to $2147483647$)</td>
<td>32-bit signed</td>
</tr>
<tr>
<td>long</td>
<td>$-2^{63}$ to $2^{63} - 1$</td>
<td>64-bit signed</td>
</tr>
<tr>
<td></td>
<td>(i.e., $-9223372036854775808$ to $9223372036854775807$)</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>Negative range: $-3.4028235E + 38$ to $-1.4E - 45$</td>
<td>32-bit IEEE 754</td>
</tr>
<tr>
<td></td>
<td>Positive range: $1.4E - 45$ to $3.4028235E + 38$</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>Negative range: $-1.7976931348623157E + 308$ to $-4.9E - 324$</td>
<td>64-bit IEEE 754</td>
</tr>
<tr>
<td></td>
<td>Positive range: $4.9E - 324$ to $1.7976931348623157E + 308$</td>
<td></td>
</tr>
</tbody>
</table>
public class TipCalculator {

    public static void main(String[] args) {
        // Set up variables
        double subTotal = 14.37;
        double tipAmount = 20.0;
        double tipTotal;
        double total;

        // Calculate tip total

        // Add tip total to food total

        // Print out result
    }
}
Mathematical Operations

• + Addition (also used for concatenating strings)
• - Subtraction
• * Multiplication
• / Division
• % Modulo (Remainder)
Mathematical Operations

• Order of operations:
  • () first,
  • * / % next, left to right,
  • + - last, left to right

• Tip! if you get confused, put more parens in to be explicit
Integer division

• when both numbers are ints, the result is also an int (so no decimal place, that part is truncated)

• So what does 1 / 2 = ?

• If you want the result to be a decimal, you need to make at least one number a decimal (double).

• 1 / 2.0 = 0.5
Modulo

- Super useful in programming
- even number % 2 is always $= 0$ and odd number % 2 is always $= 1$
Exponents

• For now, all you need to know is that Math.pow(a, b) is taking $a^b$
public class TipCalculator {

    public static void main(String[] args) {
        // Set up variables
        double subtotal = 14.37;
        double tipAmount = 20.0;
        double tipTotal;
        double total;

        // Calculate tip total
        tipTotal = subtotal * (tipAmount / 100);
        System.out.println("Tip total: "+ tipTotal);

        // Add tip total to food total
        total = subtotal + tipTotal;

        // Print out result
        System.out.println("Total bill: "+ total);
    }
}
Reading from the console using the Scanner class

- System.out is standard output device, System.in is standard input

- Console input isn't directly supported, so we can use the Scanner class to create an object that will read from System.in
Reading from the console using the Scanner class

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

• Creates an object of Scanner type, assigns its reference to the input variable.

• Has methods (like our classes do and will), like nextDouble()
Scanner class

- First time we’ll be importing from other packages

- to use a Scanner object, we need to import the package where it lives - java.util - by adding this line to the top of our program:

  ```java
  import java.util.*;
  ```

- You could also specifically import just the scanner object by using

  ```java
  import java.util.Scanner;
  ```

- to get a value from the Scanner, we use one of the next methods such as `nextInt()`, `nextDouble()`, or `next()`
Scanner class

• When we use a scanner object, we need to first prompt the user for what we’d like them to enter

• We do this by writing to the console

    System.out.print("Please enter the subtotal: ");
print vs println?

• What’s the difference between print and println?

• When might be a good idea to use each one?
Let's get input for the total and tip percentage

```java
import java.util.Scanner;

public class TipCalculator {
    public static void main(String[] args) {
        // Set up variables
        double subtotal;
        double tipAmount;
        double tipTotal;
        double total;
        Scanner input = new Scanner(System.in);

        // Read subtotal from console
        System.out.print("Enter the subtotal: ");
        subtotal = input.nextDouble();

        // Read tipAmount from console
        System.out.print("Enter the tip percentage (ex. 20.0): ");
        tipAmount = input.nextDouble();

        // Calculate tip total
        tipTotal = subtotal * (tipAmount / 100);
        System.out.println("Tip total: " + tipTotal);

        // Add tip total to food total
        total = subtotal + tipTotal;
    }
}
```
Programming challenge

- Write a program that will convert Fahrenheit to Celsius

\[ C = \left( F - 32 \right) \times \frac{5}{9} \]