Garfield AP CS

Java first impressions, expressions and variables

First impressions

- Easier or harder than you thought?
- Which parts were confusing?
- What kinds of errors did you get?
- How is a computer different from a person?

Classes recap

- All programs must be inside a class
- All runnable classes must have a main method
- The class name must match the file name
- Classes start and end with a curly brace
- The class name must be a legal identifier

```
public class ClassName {


}
```

Methods

- Group related statements together
- Create new Java commands
- Must be called to do anything
- Name must be a legal identifier

```
public static void methodName() {
    statements;
}
```

Method call:

```
methodName();
```

Legal identifiers

- Must start with a letter or _
- Capitalization counts (case-sensitive)
- No spaces

Main

- Special method Java always starts with
- Always looks the same
- Opened and closed by curly braces

```
public class ClassName {
   public static void main(String[] args) {
      <statements>;
   }
}
```

Data types

- **type**: A category or set of data values.
  - Constrains the operations that can be performed on data
  - Many languages ask the programmer to specify types
  - Examples: integer, real number, string
- Internally, computers store everything as 1s and 0s

```
104    → 01101000
"hi"   → 01101000110101
```

Java's primitive types

- **primitive types**: 8 simple types for numbers, text, etc.
  - Java also has **object types**, which we'll talk about later

Name	Description	Examples
`int`	integers	`42, -3, 0, 926394`
`double`	real numbers	`3.1, -0.25, 9.4e3`
`char`	single text characters	`'a', 'X', '?', '\n'`
`boolean`	logical values	`true, false`

  - Why does Java distinguish integers vs. real numbers?

Expressions

- **expression**: A value or operation that computes a value.
  - Examples:
    ```
    1 + 4 * 5
    (7 + 2) * 6 / 3
    42
    ```
  - The simplest expression is a *literal value*.
  - A complex expression can use operators and parentheses.

Arithmetic operators

- **operator**: Combines multiple values or expressions.

`+`	addition
`-`	subtraction (or negation)
`*`	multiplication
`/`	division
`%`	modulus (a.k.a. remainder)

- As a program runs, its expressions are *evaluated*.
  - `1 + 1` evaluates to `2`
  - `System.out.println(3 * 4);` prints `12`
    - How would we print the text `3 * 4` ?

Notes on operators

- Dividing integers results in integers
- Dividing by 0 gives an error
- % computes remainder of integer division
  - applications?
- */% have higher precedence than +-

Real numbers

- Type `double`
- Place a .0 after an `int` to get a `double`
- Mixing an `int` and a `double` results in a `double`

String concatenation

- “Glue” text together
- The result is a string
- Useful for printing out numbers

Try it

- What values result from the following expressions?
  - `9 / 5`
  - `695 % 20`
  - `7 + 6 * 5`
  - `7 * 6 + 5`
  - `248 % 100 / 5`
  - `6 * 3 - 9 / 4`
  - `(5 - 7) * 4`
  - `6 + (18 % (17 - 12))`
  - `6 * 3.4 - 2`
  - `“goo” + 9.3 / (1 + 2.0)`

Variables

- Way to store information
- Must be declared with a type
- Must be initialized

```
int foo = 10;
double bar = 20;
int baz;
baz = 10;
```