Recap: Java Static Methods

- Why methods?
  - Denote structure of a program
  - Allow code reuse to eliminate redundancy

- Syntax: defining methods and invoking methods

- Basic method design/implem. methodology
  - Top-down decomposition/bottom-up implementation

Recap: Static Method Example

- Write a program to print these figures.

Recap: Decomposition Example

- main
- egg
- teaCup
- stopSign
- hat
- eggTop
- eggBottom
- line
- stopLine
Structured Program version

// Prints several figures, with methods for structure and redundancy.
public class Figures3 {
    public static void main(String[] args) {
        egg();
        teaCup();
        stopSign();
        hat();
    }

    // Draws the top half of an egg figure.
    public static void eggTop() {
        System.out.println("  ______");
        System.out.println(" /      ");
        System.out.println("/        ");
    }

    // Draws the bottom half of an egg figure.
    public static void eggBottom() {
        System.out.println("\        /");
        System.out.println(" \______/";
    }

    // Draws a complete egg figure.
    public static void egg() {
        eggTop();
        eggBottom();
        System.out.println();
    }

    ...} // Draw a line of dashes.
    public static void line() {
        System.out.println("+--------+");
    }

    // Draws a teacup figure.
    public static void teaCup() {
        eggBottom();
        line();
        System.out.println();
    }

    // Draws a stop sign figure.
    public static void stopSign() {
        eggTop();
        System.out.println("| STOP |");
        eggBottom();
        System.out.println();
    }

    // Draws a figure that looks sort of like a hat.
    public static void hat() {
        eggTop();
        line();
    }
}

A Word about Style

- Structure your code properly
- Eliminate redundant code
- Use comments to describe code behavior
- Use spaces judiciously and consistently
- Indent properly
- Follow the naming conventions

Why Style?

- Programmers build on top of other's code all the time.
  - You shouldn't waste time deciphering what a method does.
- You should spend time on thinking or coding. You should NOT be wasting time looking for that missing closing brace.
- So code with style!

Foundational Programming Concepts

Any program you might want to write

- objects
- methods and classes
- graphics, sound, and image I/O
- arrays
- conditionals and loops
- Math
- text I/O
- primitive data types
- assignment statements

Outline

- Admin and recap
- Primitive data types
  - why data types
A computer can use multiple cells (e.g., 2 bytes) to store a value. Each memory cell has a set number of bits (usually 8 bits, or one byte; a bit can represent 2 values of 0 or 1). How many possible values can a byte represent?

Problem

- What does the number (combination) stored at a given memory location represent?

Two Example Possibilities

- 01011001 → number 89
- 01011001 → character Y

Problem: How can the computer tell what 01011001 stands for: a character Y or 89?

Type System

- type: A category or set of values and operations defined on those values.
- By specifying the type of a memory location, we know what the values represent.
- Many languages ask the programmer to specify types (e.g., integer, real number, character)

Primitive Data Types

- There are eight (simple) primitive data types in Java:
  - six numerical types
    - for mathematical calculation
  - characters
    - for text processing
  - Boolean (logical) values
    - for decision making
The differences among the various numeric primitive types are their storage sizes and representation format, and hence the ranges & precision of the values they can store.

- **Byte**: 1 byte, Min Value: -128, Max Value: 127
- **Short**: 2 bytes, Min Value: -32,768, Max Value: 32,767
- **Int**: 4 bytes, Min Value: -2,147,483,648, Max Value: 2,147,483,647
- **Long**: 8 bytes, Min Value: \(-9 \times 10^{18}\), Max Value: \(9 \times 10^{18}\)

Represented using the IEEE 754 format
- with limited # of precision bits
- See Precision.java

Java Numerical Value and Type

- **Java is a strongly typed language, i.e., every data item has a type**
- **An integer literal is by default of type int**
  - that is, a literal number 4 in Java is of type int
  - to say that the number 4 is of type long, write 4l or 4L (4L is preferred over 4l since lower case "l" is hard to distinguish from 1)
- **A real (floating point) literal (e.g., -1.23 6.12e23) is by default of type double**
  - to say that the number 0.1 is of type float, write 0.1f or 0.1F

Variable: A piece of the computer's memory that is given a name and a type to store value of the type.
- Like preset stations on a car stereo, or cell phone speed dial

Steps for using a variable:
- **Declare it**: state its name and type
- **Assign value**: initialize or update its value
- **Use it**: print it or use it as part of an expression
**Declaration**

- **Variable declaration**: Sets aside memory for storing a value.
  - Variables must be declared before they can be used.

- **Syntax**:
  - `<type> <name>`;
  - `int x;`  
  - `double myGPA;`

**Assignment**

- **Assignment**: Stores a value into a variable.
  - The value can be an expression; the variable stores its result.

- **Syntax**:
  - `<name> = <expression>`;
  - `int x;`  
  - `double myGPA;`
  - `x = 3;`
  - `myGPA = 1.0 + 2.25;`

- A variable can only store a value of its own type.

**Practice: Integer or Real Number?**

- Which category is more appropriate?

<table>
<thead>
<tr>
<th>integer</th>
<th>real number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Your locker number</td>
<td>1. Your locker number</td>
</tr>
<tr>
<td>2. The population of the world</td>
<td>2. The population of the world</td>
</tr>
<tr>
<td>3. Your grade point average</td>
<td>3. Your grade point average</td>
</tr>
<tr>
<td>4. A person's height in meters</td>
<td>4. A person's height in meters</td>
</tr>
<tr>
<td>5. The value of pi</td>
<td>5. The value of pi</td>
</tr>
</tbody>
</table>

**Questions**

- Question: to represent the number of students at Yale, which numeric data type variable do you use?
- Question: to represent the world population, which numeric data type variable do you use?
- Question: to represent pi as 3.14159265359, which numeric data type variable do you use?

**Real Life Example: Ariane 5**

- Historical example: Ariane 5 explosion in 1996
  - (http://www.youtube.com/watch?v=kYUrqdUyEpI; http://www.imr.umn.edu/~arnold/disasters/ariane.html)
Real Life Example: Ariane 5

- Historical example: Ariane 5 explosion in 1996
  (http://www.youtube.com/watch?v=kYUqduYEpE)
  (http://www.ima.umn.edu/~arnold/disasters/ariane.html)

- Reason: range error
  - trying to store a 64-bit real number (a double) to a 16-bit integer led to the crash

Real Life Example: Patriot Failure

- The Patriot Missile Failure in 1991
  - Perfect detection of a Scud missile, but the intercepting Patriot missed the target
  - Reason: precision error
    - a computer cannot represent 0.1 precisely; for a 24-bit floating point number they used, it is off by 0.0000000095.
    - After 100 hours in operation, it is off by 0.34 seconds (0.0000000095*100 hours * 60 min/hour * 60 sec/min * 10), leading to an error of about 600 meters
  (http://www.ima.umn.edu/~arnold/disasters/patriot.html)

In the Movie

- http://www.youtube.com/watch?v=G_wiXgRWrIU

Characters

- A char is a single character from a character set
- A character set is an ordered list of characters; each character is given a unique number
- Character literals are represented in a program by delimiting with single quotes:
  'a' 'X' '7' '§' ',' '
'

Java Character Set

- Java uses the Unicode character set, a superset of ASCII
  - uses sixteen bits (2 bytes) per character, allowing for 65,536 unique characters
  - it is an international character set, containing symbols and characters from many languages
  - code chart can be found at:
    http://www.unicode.org/charts/

Boolean

- A boolean value represents logical value: true or false
- The keywords true and false are the only valid values for a boolean type
- A boolean can also be used to represent any two states, such as a light bulb being on or off
Outline

- Admin and recap
- Primitive data types
  - storage and representation
  - operations

Data Type and Operations

- A type defines not only the storage/representation but also the allowed and meaning of operations
  - Discussions: reasonable operations that can be performed on two operands
    - Integers: i1 ? i2
    - Strings: s1 ? s2
    - Characters: c1 ? c2

Data Type and Operations

<table>
<thead>
<tr>
<th>type</th>
<th>set of values</th>
<th>literal values</th>
<th>operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>characters</td>
<td>'A'</td>
<td>compare (more details later on ~)</td>
</tr>
<tr>
<td>String</td>
<td>sequences of characters</td>
<td>&quot;hello&quot;</td>
<td>concatenate +</td>
</tr>
<tr>
<td>int</td>
<td>integers</td>
<td>17</td>
<td>compare, add +, sub -, multiply *, divide /, modulus %</td>
</tr>
<tr>
<td>double</td>
<td>floating-point numbers</td>
<td>3.1415</td>
<td>compare, add +, sub -, multiply *, divide /, modulus %</td>
</tr>
<tr>
<td>boolean</td>
<td>truth values</td>
<td>true</td>
<td>==, !=, and &amp;&amp;, or</td>
</tr>
</tbody>
</table>

Interpretation

You should think that there are multiple versions of the same operator, each for a type, e.g.,
- +int +string ...
- /int /double ...

Integer Division with /

- When we divide integers, the result is an integer (the fractional part is discarded)
  - 14 / 4 is 3, not 3.5
  - 12 / 3 is 4

- More examples:
  - 32 / 5 is 6
  - 8 / 10 is 0
  - 156 / 100 is 1

- Dividing by 0 causes an error when your program runs.
### Integer Remainder with \% 

- The % operator computes the remainder from integer division.
  - $14 \div 4$ is 2
  - $218 \div 5$ is 3

<table>
<thead>
<tr>
<th>Integer</th>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>218</td>
<td>3</td>
</tr>
</tbody>
</table>

**Practice:**

- $45 \% 6$ is 3
- $2 \% 2$ is 0
- $8 \% 20$ is 8
- $11 \% 0$ is 11

- Obtain last digit of a number: $230857 \% 10$ is 7
- Obtain last 4 digits: $230857 \% 10000$ is 857
- See whether a number is odd: $7 \% 2$ is 1, $42 \% 2$ is 0

### Evaluating Arithmetic Expression 

- **Arithmetic operators can be combined into complex arithmetic expressions**
  - $(7 + 2) \times 6 \div 3$

- The evaluation order of the operators in an arithmetic expression is determined by a well-defined precedence order
  - **Remember?**
    - Pretty Please My Dear Aunt Sally

### Operator Precedence Rules 

- Generally operators evaluate left-to-right:
  - $1 - 2 - 3$ is $1 - (2 - 3)$ which is 2

- But $\times / \div$ have a higher level of precedence than $+$ -
  - $1 - 3 \times 4$ is $-11$

- Parentheses can force a certain order of evaluation:
  - $(1 + 3) \times 4$ is 16

- Spacing does not affect order of evaluation:
  - $1 + 3 \times 4 - 2$ is 11

### Precedence: Examples 

- What is the order of evaluation in the following expressions?
  - $a + b + c + d + e$
  - $a + b + c + d \div e$
  - $a \div (b + c) - d \div e$
  - $a \div (b + (c + (d - e)))$

- $1 * 2 + 3 * 5 \% 4$
- $1 + 8 \div 3 * 2 - 9$
- $2 + 3 * 5 \% 4$
- $1 + 2 * 2 - 9$
- $2 + 15 \% 4$
- $1 + 4 - 9$
- $2 + 3$
- $5 - 9$
- $-4$
Practice: Precedence Questions

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))

Real Number Example

\[ 2.0 \times 2.4 + 2.25 \times 4.0 / 2.0 \]

\[ 4.8 + 2.25 \times 4.0 / 2.0 \]

\[ 4.8 + 9.0 / 2.0 \]

\[ 4.8 + 4.5 \]

\[ 9.3 \]

Outline

- Admin and recap
- Primitive data types
  - storage and representation
  - operations
  - expressions
  - mixed-type operations

Problem

Sometimes it is more efficient and natural to represent data as one type, but during a computation, we may want to get desired result in a different type.

- e.g. raw grade points and # of grades as integers, but GPA as double (see GPA.java)

Sometimes we just write mixed-type expressions

- 4.0 / 8 (Q: is the value 0 or 0.5?)

Data Conversion

Data conversion is the conversion of data from one type to a different type, e.g.,

- int → double,
- double → int,
- int → string
**Data Conversion**

- Conversion is **per-operator**, occurring when the operator is evaluated according to the precedence rule.

- Java tries a set of predefined data conversion rules:
  - If successful, you get the results.
  - If not, you get a compiler error.

---

**Data Conversion Rule: Arithmetic (numeric) Promotion**

- Occurs **automatically** when the operands of a binary arithmetic operator are of different types:
  - if either operand is `double`, the other is converted to `double`
  - otherwise, if either operand is `float`, the other is converted to `float`
  - otherwise, if either operand is `long`, the other is converted to `long`
  - otherwise, both operands are converted to `int`

Examples:
- `4.0 / 8` (which is it: `double`, `float`, `int`)
- `4 / 8.0` (which is it: `double`, `float`, `int`)
- `4 / 8` (which is it: `double`, `float`, `int`)

---