CMSC 131
Object-Oriented Programming I
Precedence, Short Circuiting, Casting,
Static Methods
Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr,
Fawzi Emad, David Mount, Jan Plane
Overview

- Precedence
- Short Circuiting
- Casting
- Static Methods
Precedence

- Explains how to evaluate expressions
  - What is value of $1 - 2 + 3 * 4$?
  - **Precedence rules** answer this question
    - Higher-precedence operators evaluated first
    - Example from math: “Please, Excuse my Dear Aunt Sally” or PEMDAS
      - Multiple and divide (higher precedence) before you add and subtract (lower precedence)
  - Java follows “Aunt Sally’s Rules” ... but what about other operators?
Java Precedence Rules

- **parentheses**: ( )
- **unary ops**: +x, -x, ++x, --x, x++, x-- (!x)
- **multiply/divide**: *, /, %
- **add/subtract**: +, -
- **comparisons**: <, >, <=, >=
- **equality**: ==, !=
- **logical and**: &&
- **logical or**: ||
- **assignments**: =, +=, *=, /=, %= (these are right to left associative)

Higher precedence on top
Examples

- \(x \times y + -z\)
  
  Same as \((x \times y) + (-z)\)

- \((x \leq y \&\& y \leq z \; || \; w > z)\)
  
  Same as \(((x \leq y) \&\& (y \leq z)) \; || \; (w > z)\)

- What is value of \(1 - 2 + 3 \times 4?\)
  
  \[
  = 1 - 2 + 3 \times 4 \\
  = 1 - 2 + (3 \times 4) \\
  = (1 - 2) + 12 \\
  = -1 + 12 \\
  = 11
  \]
Should You Rely on Precedence?

- No!
- The only ones people can remember are
  - “Please Excuse My Dear Aunt Sally” (PEMDAS)
  - And maybe unary and increment/decrement operators

- Bad:
  
  ```
  if (2 * x++ < 5 * z + 3 && -w != x / 2)
  ```

- Better:
  
  ```
  if ((2 * x++ < 5 * z + 3) && (-w != x / 2))
  ```

- Best:
  
  ```
  if (((2 * x++) < (5 * z + 3)) && (-w != (x / 2))
  ```
Strive for Readable Code

- **Alternative #1**
  
  ```java
  if ((temp > 98 && temp <= 100) || (systolic <= 120 && diastolic < 80))..
  ```

- **Alternative #2**
  
  ```java
  boolean tempIsOK = (.....)
  boolean BPIsOK = (....)
  if (tempIsOK || BPIsOK)....
  ```
As soon as Java knows an answer → it quits evaluating the expression

What does Java print?
```
int x = 0, y = 1;
if ((y > 1) && (++x == 0)){
   --y;
}
System.out.println(x);
=> 0
```

Why?
- `y > 1` is false
- The result of `&&` will be false, regardless of second expression
- Java therefore does not evaluate second expression of `&&`

This treatment of `&&`, `||` is called **short-circuiting**
- Subexpressions evaluated from left to right
- Evaluation stops when value of over-all expression is determined

**Example:** ShortCircuiting.java
Examples

- What does Java print?
  ```java
  int x = 0, y = 1;
  if ((y >= 1) && (++x == 0)) {
      --y;
  }
  System.out.println(x);
  1
  ```

- What does Java print?
  ```java
  int x = 0, y = 1;
  if ( ((y > 1) && (++x == 0)) ||
       ((y == 1) && (x++ == 0)) ) {
      --y;
  }
  System.out.println(x);
  System.out.println(y);
  1
  0
  ```
What does Java print?

```java
int x = 0, y = 0;
while (x++ <= 4){
    y += x;
}
System.out.println (y);
=> 15
```
Programming with Side-Effects

Generally

- Side effects in conditions are hard to understand
- Good programming practice
  - Conditions should be side-effect-free
  - Side effects should be in “stand-alone statements”
- Major Goal ➔ Strive to create the most readable and maintainable code.
Primitive Types and their Hierarchy

- double
- float
- long
- int
- short
- byte

```java
int x = 7.2;
double y = 6;
```

- Changing to something else Further Up this list is acceptable
  - called “Widening Conversion”
- Changing to Something else Further Down this list is not acceptable
  - called “Narrowing Conversion”
- **Explicit casting** needed for when you want to go lower in the list
Type Casting - implicit

Which of the following are legal?

- `int x = 3.5;`
  **Illegal**: `3.5` is not an `int`

- `float x = 3;`
  **Legal**: `3` is an `int`, which is also a `float`

- `long i = 3;`
  **Legal**: `3` is an `int`, which is also a `long`

- `byte x = 155;`
  **Illegal**: `155` is too big to be a `byte` (> 127)

- `double d = 3.14159F;`
  **Legal**: `3.14159F` is a `float`, which is also a `double`
Mixed Expressions with Explicit Type Casting

- **What is result of**
  
  ```java
  float x = 3 / 4;
  ```
  
  - x assigned value `0.0F`
  - Why?
    - 3, 4 are ints
    - So integer / operation is used, yielding 0, before upcasting is performed

- **To get floating point result, use explicit casting**
  
  ```java
  float x = (float) 3 / (float) 4;
  ```
  
  Assigns x the value `0.75F`

- **Can also do following**
  
  ```java
  float x = (float) 3 / 4;
  ```
  
  Why?
  - `(float) 3` returns a value type `float (3.0F)`
  - 4 is an int
  - In this case, Java compiler uses widening conversion on “lower” type (here, `int`) to obtain values in same type before computing operation

- **Or:**
  
  ```java
  float x = 3.0f / 4;
  ```
main method

public static void main(String args[]){
    // statements here
}

- All projects and examples have defined this method
- No explicit call needed
- Parts of the line
  - Name → main
  - Parameter List → String args[]
  - Return type → void
  - Access → public (more on this later)
  - Modifier → static
Non-main static public methods: defining, invoking and commenting

- Defined based on a name and a list of parameters

  ```java
  public static void name(parameterlist){
  body
  }
  ```

- Invoked by stating its name and giving an argument for each element of the parameter list

  ```java
  name(argumentlist);
  ```

- **parameter list**
  - type name for each item in the list
  - e.g. (MyGrid grid, char where)

- **argument list**
  - expression for each item in the list
  - e.g. (grid, 't')

- Each method must have a well defined purpose
  - That information goes into a comment before the method definition
  - Each parameter’s purpose should be explained
  - Return value’s purpose should be explained
Static methods

- A static method is associated with a class
  - Not an individual instance (object)
- Must have all of the same parts as the main method
  - `public/private static returnType name(argList){
    body
  }

- **Example:** Driver.java, Triangle.java
- Notice that in Java we can have multiple classes and one can refer to the other if they are in the same location (package)
- What happens when a method is called?
- `private` vs. `public`
Method information: parameters and arguments

- **Parameter list**
  - type name
  - e.g. (MyGrid grid, char where)

- **Argument list**
  - expression
  - e.g. (grid, ‘t’)

Matched between the arguments and the parameters based on position in the list.