Today

• Manipulating primitive data types
• Eclipse & submitting projects
Short Questions?
Quiz

Quiz grading:

• Most questions: equal grade weight is given to participation and correctness

• Some questions are graded by participation only

• Procedure update: individual quizzes will be added to the D2L grade book on an individual basis
  • The 15% of final grade will be distributed evenly across all quizzes that we take
  • Drop the lowest two
Example: “=” operator

int foo;
foo = 5;
foo = foo + 3;
System.out.println(foo);

“=” is about storage, not equality!
Juggling Exercise
Handing In a Project

Process:
• Write, test and debug the code
• Export project to a Zip file
• Submit to D2L dropbox
Exporting a Project

• Select the project in the Package Explorer
• File: Export
• Export destination: General: Double click on “Archive File”
• To archive file: Give the name of the zip file
  • Note: you may also have to browse to a destination folder
• Leave “Save in zip format” selected
• Click Finish
Mathematical Operators

• Satisfy standard precedence relationships:
  • Level 2:  ++   --
  • Level 3:  ( )  for grouping of expressions
  • Level 4:  *   /   %
  • Level 5:  +   -

• Each operator is potentially defined differently for different data types
Some Syntactic Notes

Curly brackets {} and parentheses () always come in matching pairs
• {}: used to group several statements together
• (): used for method (or function) definition/calls
• Eclipse helps you to keep track of these pairs by:
  • Indenting code within {}
  • Giving errors when one of a pair is missing

Semicolons (;) are necessary to end a single code statement.
• Eclipse will also give you an error if you have forgotten one
Camel Case Convention

• We try to make our identifiers as descriptive as possible by describing them with multiple words
• However, a space character cannot be used as part of an identifier
• So, we cram the words together:

  ```
  int numberOfCamels;
  ```

• Note:
  • First letter of a variable name is (by convention) lower case
  • But the first letter of a class name is upper case
final keyword

• Some of our variables are not actually variable – they are constants
• One could just include the value in the code, but these “magic values” are not very descriptive & make the code hard to read and maintain
• Instead, we want to use a descriptive identifier, but we want the compiler to enforce the fact that it will not change
• Convention is that these constants use only capital letters

final double CM_PER_INCH = 2.54;
Characters

• An individual character is stored in a single byte
• Since a byte is just a number, we must have some way of mapping numbers to glyphs (the visual representation of a character)
ASCII Encoding of Characters

- This encoding served us well for many years
- But, we really want to be able to represent any glyph
- Answer: Unicode uses multiple bytes to capture a single character (the number of bytes depends on the standard)
Special Characters

• ‘\n’ is a single character that means “new line” (and is often implemented as both a “new line” and a “carriage return”)
• ‘\t’ is a tab
• ‘\\’ is a \
Mixing Types with Operators

int foo = 4;
double bar = 5.3;
System.out.println(foo + bar);

• For the + operation: the value 4 is first converted to a double; then, it is added to 5.3
• The result (a double) is then converted to a string for use by println()
Mixing Types with Operators

• Not all conversions are automatic (in fact, few are)
  • A double will not be converted automatically to an int

• We tell the Java compiler that such a conversion is allowed through casting:

```java
int foo;
double bar = 5.3;

foo = (int) bar;
System.out.println(foo);
```

Casts are rare. If you think you need it, something might be wrong in your design or implementation.
Wrap Up

Due this week:
• HW 1: Turing’s Craft
• Project 0: Eclipse + D2L

Also out:
• HW 2: due next week

Next time:
• Conditionals