SE-1010 Software Development 1

Integers (Part 2)

Part II – The Revenge

Dr. Walter W. Schilling, Jr.

Instructor

Minute Quiz

- Four integer-like data types within Java are byte, short, int, and long.
- A short integer data type requires 16 bits to store a value.
- The % operator performs a remainder (modulus) operation.
- Overflow occurs if two large numbers are added together, exceeding the capacity of the data type. (Rollover)

Homework

- 7-30, 7-33

Precedence

- 2 + 3 * 4 → 14
- Is this 20 or 14?

2nd

1st

3*4 ⇒ 12

2 + 12 ⇒ 14 (Wow!)

3 Precedence Rules to worry about – (Hint: More will be coming next week!)

- Rule 1:
  - {*, /, %} are performed before {+,-}.
- Rule 2:
  - When two operators have equal precedence, the *leftmost* one is performed first.
- Rule 3:
  - An expression inside parentheses is evaluated on its own before being used by operands outside the parentheses.

Solve the following:

•2 - 4 / 3 + 2 $\quad \frac{2}{3}$ $\cancel{\frac{1}{3}}$

2 − 1 + 2

3

•2 – 4 / (3 + 2) $\quad$ 2

2 − 4 / 5

2 − 0 ⇒ 2

•17 / 8 / 3 $\quad$ 0

2 / 3 ⇒ 0

•17 / (8 / 3)

8/3

⇓

17 / 2 ⇒ 8

Best Practice

- If you are in doubt about precedence, always explicitly parenthesize your source code.
  - ~~Won't hurt if you do not need them~~
  - ~~But,~~
  - May avoid a costly bug if you are wrong in interpreting precedence.

Shortcuts

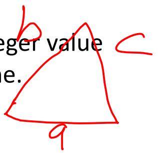
Adds a value of 27 to the variable n, storing the result back in n.

expression	*shortcut*
`n = n + 27;`	`n += 27;`
`n = n - 5;`	`n -= 5;` — Subtraction
`n = n * 3;`	`n *= 3;` — multiplication
`n = n / 2;`	`n /= 2;` — Divides
`n = n + 1;` n+=1;	`n++;`
`n = n - 1;`	`n--;`

DO NOT use more than one of these on a line of code.

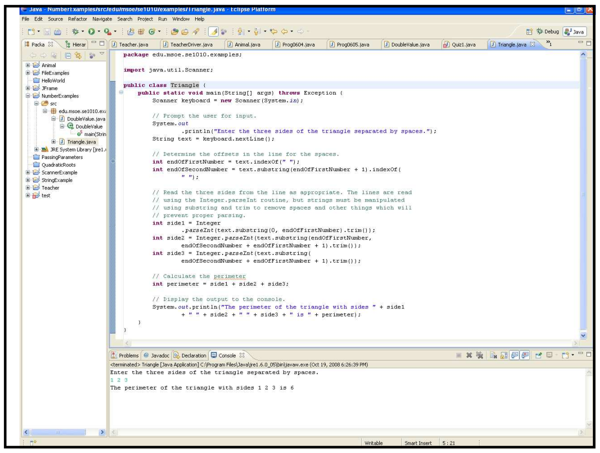
Reading ints from Strings

- We can parse a String for an integer value using the Integer.parseInt routine.



- Problem solving example…
  - Lets calculate the perimeter of a triangle by adding the three sides together.
  - 3 sides are to be added on the same line separated by spaces.

a b b c ⇒ perimeter
a+b+c



```
package edu.msoe.se1010.examples;

import java.util.Scanner;

public class Triangle {
    public static void main(String[] args) throws Exception {
        Scanner keyboard = new Scanner(System.in);

        // Prompt the user for input.
        System.out
                .println("Enter the three sides of the triangle separated by spaces.");
        String text = keyboard.nextLine();

        // Determine the offsets in the line for the spaces.
        int endOfFirstNumber = text.indexOf(" ");
        int endOfSecondNumber = text.substring(endOfFirstNumber + 1).indexOf(
                " ");

        // Read the three sides from the line as appropriate. The lines are read
        // using the Integer.parseInt routine, but strings must be manipulated
        // using substring and trim to remove spaces and other things which will
        // prevent proper parsing.
        int side1 = Integer
                .parseInt(text.substring(0, endOfFirstNumber).trim());
        int side2 = Integer.parseInt(text.substring(endOfFirstNumber,
                endOfSecondNumber + endOfFirstNumber + 1).trim());
        int side3 = Integer.parseInt(text.substring(
                endOfSecondNumber + endOfFirstNumber + 1).trim());

        // Calculate the perimeter
        int perimeter = side1 + side2 + side3;

        // Display the output to the console.
        System.out.println("The perimeter of the triangle with sides " + side1
                + " " + side2 + " " + side3 + " is " + perimeter);
    }
}
```

```
<terminated> Triangle [Java Application] C:\Program Files\Java\jre1.6.0_05\bin\javaw.exe (Oct 19, 2008 6:26:39 PM)
Enter the three sides of the triangle separated by spaces.
1 2 3
The perimeter of the triangle with sides 1 2 3 is 6
```

Java Also Provides

- Byte.parseByte(String) — -128 to +127
- Short.parseShort(String) — -32768 to 32767
- Long.parseLong(String) — -2^{63} to 2^{63}

Objects versus primitive data types

new Integer(6) ⇒ Integer class instance w/ value of 6. calls the constructor

Objects	**Primitive Datatypes**
Created by **new** and a constructor – new Animal("fido", "woof")	Created by giving an expression or a literal: – 37 int x = 37;
Variables are *references* to objects a	Variables hold the value itself a 38
Have internal structure — instance variables (attributes of the object)	Have no internal structure.
Do work by receiving messages (methods):	Can be combined into expressions using primitive operators: c = a+b d = a + 10;
Are correlated with the problem our program is trying to solve.	Are correlated with the storage elements computer hardware.
We can define new classes of objects (a toString)	We *cannot* define new primitive datatypes. int ⇒ 32 bits of memory

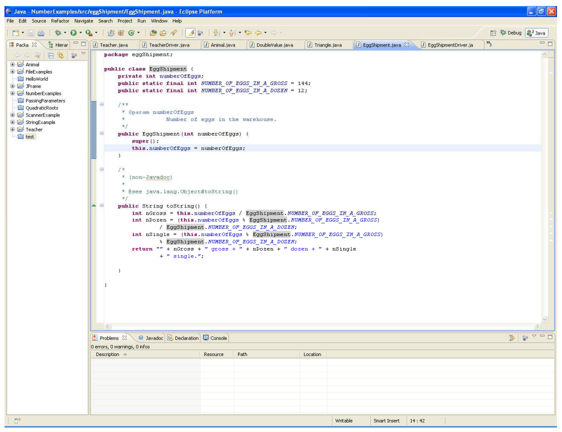
Converting ints to Strings

- What you want to do:
  - int i = 17;
  - String s = i;
- Legal way of doing it:
  - int i = 17;
  - String s = "" + i;

→ Compiler error
~~int~~ String Strongly typed
Nature of Java forbids this.
Concat operation
null string
'i' "17"

Problem Solving

- The egg crate problem
  - Create a class which
  - 1. Stores the number of eggs in a warehouse
  - 2. Can compute the number of gross (144) eggs within the warehouse, the number of dozen in the warehouse, and the remaining single eggs.


Java - NumberExamples/src/eggShipment/EggShipment.java - Eclipse Platform

```
package eggShipment;

public class EggShipment {
    private int numberOfEggs;
    public static final int NUMBER_OF_EGGS_IN_A_GROSS = 144;
    public static final int NUMBER_OF_EGGS_IN_A_DOZEN = 12;

    /**
     * @param numberOfEggs
     *            Number of eggs in the warehouse.
     */
    public EggShipment(int numberOfEggs) {
        super();
        this.numberOfEggs = numberOfEggs;
    }

    /*
     * (non-Javadoc)
     *
     * @see java.lang.Object#toString()
     */
    public String toString() {
        int nGross = this.numberOfEggs / EggShipment.NUMBER_OF_EGGS_IN_A_GROSS;
        int nDozen = (this.numberOfEggs % EggShipment.NUMBER_OF_EGGS_IN_A_GROSS)
                / EggShipment.NUMBER_OF_EGGS_IN_A_DOZEN;
        int nSingle = (this.numberOfEggs % EggShipment.NUMBER_OF_EGGS_IN_A_GROSS)
                % EggShipment.NUMBER_OF_EGGS_IN_A_DOZEN;
        return "" + nGross + " gross + " + nDozen + " dozen + " + nSingle
                + " single.";

    }

}
```

0 errors, 0 warnings, 0 infos

Problem

- I want to control the format of how Java ints are printed to the console
- Mechanism:
  - System.out.printf
  - String.format

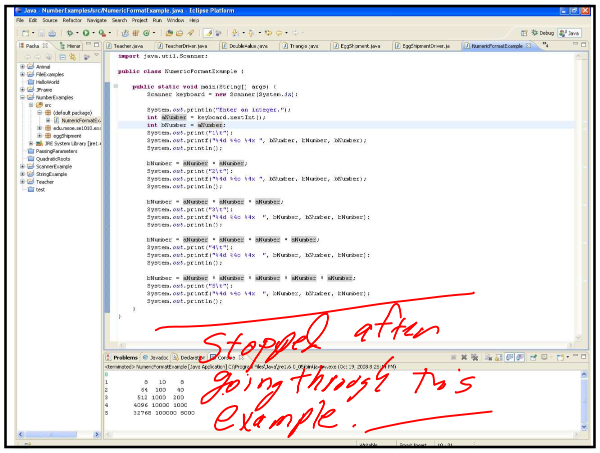
formats a printstream using given parameters
formats a string

- **printf("1 %4d\n", n)**
- **"1 %4d\n"** is a *control string*
- **%4d** means right-aligned in a field of width 4
- After the control string come the variables to print.

Generic #
Print a decimal Number 4 characters wide
bbb1
bb10
10

Format specifies

- %d signed int signed decimal integer — Most common
- %o unsigned int unsigned octal integer — octal number (Base 8 Number)
- %x, %X unsigned int unsigned hexadecimal integer, lowercase or uppercase — Hex ex 53BA
- %f float real number, standard notation %e,
- %E float real number, scientific notation (lowercase or uppercase exponent marker) %g,
- %G float same format as %f or %e, depending on the value.
  - Scientific notation is used only if the exponent is greater than the precision or less than -4.
- %s String string
- %c char character



```
import java.util.Scanner;

public class NumericFormatExample {

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);

        System.out.println("Enter an integer.");
        int aNumber = keyboard.nextInt();
        int bNumber = aNumber;
        System.out.print("1\t");
        System.out.printf("%4d %4o %4x ", bNumber, bNumber, bNumber);
        System.out.println();

        bNumber = aNumber * aNumber;
        System.out.print("2\t");
        System.out.printf("%4d %4o %4x ", bNumber, bNumber, bNumber);
        System.out.println();

        bNumber = aNumber * aNumber * aNumber;
        System.out.print("3\t");
        System.out.printf("%4d %4o %4x  ", bNumber, bNumber, bNumber);
        System.out.println();

        bNumber = aNumber * aNumber * aNumber * aNumber;
        System.out.print("4\t");
        System.out.printf("%4d %4o %4x  ", bNumber, bNumber, bNumber);
        System.out.println();

        bNumber = aNumber * aNumber * aNumber * aNumber * aNumber;
        System.out.print("5\t");
        System.out.printf("%4d %4o %4x  ", bNumber, bNumber, bNumber);
        System.out.println();
    }
}
```

```
<terminated> NumericFormatExample [Java Application] C:\Program Files\Java\jre1.6.0_05\bin\javaw.exe (Oct 19, 2008 8:26:14 PM)
8
1          8    10     8
2         64   100    40
3        512  1000   200
4       4096 10000  1000
5      32768 100000 8000
```

Stopped after going through this example.

String.format

- Similar to printf, except that instead o printing, a string is returned.
- String.format("The square of%3d is%4d.",
  25, 625)
- Returns
  - `"The square of 25 is 625."`

20

Recommended Reading

- We didn't get to it in class, but
- READ SECTION 7.10 and 7.11
  - Excellent example of problem solving with integers and complex numbers.

Integers 21