Errors
(Using the Scanner and String Classes)
Anatomy of a Java Program
Lecture 9

Some slides borrowed from Kurt Eiselt, Tamara Munzner, and Steve Wolfman. Some learning goals from Beth Simon.

Readings
This Week’s Reading: Ch 3.1-3.8
(Major conceptual jump…)

Labs and Tutorials
This week is Lab #3.

Midterms – Save the Dates!
- Midterm #1 is 5:30-6:30pm on February 10 (Tuesday) in Woodward IRC 2
- Midterm #2 is 6-7pm on March 11 (Wednesday) in Woodward IRC 2

If you have a conflict with the first midterm, send me an email with your name, student ID number, and a brief explanation of your conflict, by noon, Tuesday, February 3!

Extra Credit Survey
- Dr. Ben Yu is studying attitudes towards learning in introductory CS classes.
- Three surveys to complete during the term.
- Survey 1 is on WebCT now. Due January 28
- These are completely optional.

However, to encourage participation, I will give 1 pt of extra credit, applied to your labs, if you complete all three surveys in time.

PeerWise Instructions
- For people participating in PeerWise, instructions are now available on WebCT.
- Your first questions/answers must be done by February 9.
- If you have questions/problems, please contact Paul Denny.
Learning Goals
By the end of the next several lectures you will be able to...

- Explain the concepts of “class”, “object”, and “method” as used in computer programming.
- Find pre-defined classes in the Java library, understand the documentation, and use the methods for basic classes.
- Write programs that use common Java classes like BigInteger, String, and Scanner.

Learning Goals
By the end of class today you will be able to...

- Explain what syntax, runtime, and logical errors are.
- Write programs using (more features of) the Scanner and String classes.
- Navigate around a larger Java program and recognize basic building blocks, such as the class header and definition, methods definitions (bodies), and statements.

Change of Gears...

Program Development

- Use an editor to create your Java program
  - often called source code
  - code used interchangeably with program or instructions in the computer world
- Another program, a compiler or an interpreter, translates source code into target language or object code, which is often machine language
- Finally, your computer can execute object code

Syntax

- Rules to dictate how statements are constructed.
  - Example: open bracket needs matching close bracket
  - If program is not syntactically correct, cannot be translated by compiler
  - Different than humans dealing with natural languages like English. Consider statement with incorrect syntax (grammar)
    
    for weeks, rained in Vancouver it hasn’t
    
    we still have pretty good shot at figuring out meaning

Semantics

- What will happen when statement is executed
- Programming languages have well-defined semantics, no ambiguity
- Different than natural languages like English. Consider statement:
  
  Mary counted on her computer.
  
  How could we interpret this?

- Programming languages cannot allow for such ambiguities or computer would not know which interpretation to execute
Errors

- Computers follow our instructions exactly
- If program produces the wrong result, it’s the programmer’s fault
  - unless the user inputs incorrect data
  - then cannot expect program to output correct results: “Garbage in, garbage out” (GIGO)
- Debugging: process of finding and correcting errors
  - Unfortunately can be very time consuming!

Errors

- Error at compile time (during translation)
  - you did not follow syntax rules that say how Java elements must be combined to form valid Java statements

Errors

- Error at run time (during execution)
  - Source code compiles
    - Syntactically (structurally) correct
    - But program tried something computers cannot do
      - like divide a number by zero.
    - Typically program will crash: halt prematurely

What about String?

- Is String a primitive type? Is it a class?
  - String is a class, but it’s a special class!
    - Automatically imported
    - Built-in literals, e.g., “This is a String literal.”
    - + operator for concatenation
  - But it also has many other methods, that you can call, just like for any ordinary class…
    (Let’s take a look…)

Questions?
Some Available String Methods

```java
public String toUpperCase();
```
Returns a new String object identical to this object but with all the characters converted to upper case.

```java
public int length();
```
Returns the number of characters in this String object.

```java
public char charAt(int index);
```
Returns the character at the given index. Note that the first character in the string is at index 0.

More String Methods

```java
public String replace(char oldChar, char newChar);
```
Returns a new String object where all instances of oldChar have been changed into newChar.

```java
public String substring(int beginIndex);
```
Returns new String object starting from beginIndex position

```java
public String substring(int beginIndex, int endIndex);
```
Returns new String object starting from beginIndex position and ending at endIndex position

Keyboard Input

- Want to type on keyboard and have Java program read in what we type
- Store it in variable to use later
- Want class to do this
  - Build our own?
  - Find existing standard Java class library?
  - Find existing library distributed by somebody else?
- Scanner class does the trick
  - java.util.Scanner
  - Niceter than System.in, the analog of System.out

Scanner Class Example

```java
import java.util.Scanner;
public class Echo
{
    public static void main (String[] args)
    {
        String message;
        Scanner scan = new Scanner (System.in);
        System.out.println ("Enter a line of text: ");
        message = scan.nextLine();
        System.out.println ("You entered: " + message + "]");
    }
}
```

Import Scanner class from java.util package

- Declare string variable to store what user types in
import java.util.Scanner;
public class Echo {
    public static void main (String[] args) {
        String message = new Scanner(System.in).nextLine();
        String result = message + System.out.println("You entered: \\
                + message + \\");
    }
}

- Use Scanner constructor method to create new Scanner object named scan
  - could be named anything, like keyboardStuff or foo

- Prompt user for input

- nextLine method reads all input until end of line
  - returns it as one long string of characters

Scanner Class Methods
- The Scanner class has other methods to read other kinds of input, e.g.,
  - nextDouble()
  - nextInt()
  - next()
- See online documentation or section 4.7 in your book for more.

Questions?
Structure of a Java Program

- Complex things have internal structure: books, countries, humans, etc.
- Java programs can be very complex!
- Java provides a lot of tools to help structure big programs.

Structure of a Java Program

- A Java program consists of a number of classes.
  - Each class lives in a file with the same name (.java)
  - (In really large programs, classes can be grouped into packages.)
  - Each of these class files “declares” or “defines” a new data type (a class).
  - Each class usually groups together parts of a program to represent some concept or logical part of the program.
  - A class starts with a class header and then stuff enclosed in curly braces {}
Structure of a Java Program

- A Java program consists of a number of **classes**.
- Each class can contain one or more **methods**.
- A method contains a sequence of **statements**: little steps that do what is needed to make the method happen.

A Simple Java Program

```java
// Our first Java program.
/* Traditionally, one's first program in a new language prints out "Hello, World!"
 */
class HelloTester {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Questions?

A bigger Java programs…