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

Lecture 6

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.

---

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 follows 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**
- **compile-time error**
- **run-time error**
- **logical error**

**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

**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

**Logical error**
- Source code compiles
- Object code runs
- But program may still produce incorrect results because logic of your program is incorrect
  - Typically hardest problems to find

**Questions?**
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…)

Some Available String Methods

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

- public int length():
  Returns the number of characters in this String object.

- 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

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

- public String substring(int beginIndex):
  Returns new String object starting from beginIndex position

- public String substring(int beginIndex, int endIndex): Returns new String object starting from beginIndex position and ending at endIndex position

  substring(4, 7)    “o K”

H e l l o K e r m i t F r o g

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

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
  - nicer than System.in, the analog of System.out

Scanner Class Example

```java
import java.util.Scanner;
import java.util.Scanner;

public class Echo {
  public 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 + \"\\nI entered: \"
    + message + \"\\nI entered: \"
    + message + \"\\nI entered: \"
    + message + \"\\nI entered: \"
    + message + \"");
  }
}
```

Scanner Class Example

```java
import java.util.Scanner;

public class Echo {
  public 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 + \"\\nI entered: \"
    + message + \"\\nI entered: \"
    + message + \"\\nI entered: \"
    + message + \"\\nI entered: \"
    + message + \"");
  }
}
```

- Import Scanner class from java.util package
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 + \\
"\n");
}
}

- Declare string variable to store what user types in

- 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

- Print out the message on the display

- 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.
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.

A Java program consists of a number of classes.
Each class can contain one or more methods.
A method groups together some instructions to do some meaningful task for the class.
A method starts with a method header and then some stuff in curly braces {}.
(Nota that again, we are using the same word for when we create a method and for when we use a method.)
There can be a special main method. If the class is run by the Java interpreter, then execution (fetch-decode-execute) starts with the main method.

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 { }

A Simple Java Program

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

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 { }
A Simple Java Program

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

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**:
  - Execution starts with the first statement, and goes step-by-step from there. (Fetch-decode-execute)
  - We’ll learn ways to change that later.
  - Simple statements end with semi-colons. (We’ll see more complicated statements later.)

Structure of a Java Program

- A Java program consists of a number of **classes**: the things, like X-Men.
- Each class can contain one or more **methods**: what they can do, their powers
- A method contains a sequence of **statements**: how they do it.

A bigger Java programs...
Questions?

From Using to Creating...
- Like the difference between buying clothes versus designing and sewing clothes:
  - Buyer: Browses stores, looks for the clothes they want, gets the clothes, and wears them.
  - Designer/Tailor: Tries to design clothing that people want, creates patterns for making the clothing, makes lots of clothes.
  - (Designer/Tailor is also a buyer: buying fabric, thread, buttons, beading, etc. Using simpler objects to create new objects.)

From Using to Creating...
- Like the difference between buying clothes versus designing and sewing clothes:
  - Using Classes: Browse Java library, look for the classes we want, import them, and use them.
  - Creating Classes: Try to design classes that people want, create patterns for making objects, make objects.
  - (Creator is also a user: we'll often use objects when creating new objects.)

Wait! We’ve been doing this already!

A Simple Java Program
// Our first Java program.
/* Traditionally, one’s first program in a new language prints out “Hello, World!”
 */
public class HelloWorld {
public static void main(String[] args) {
    System.out.println("Hello, World!");
}
}
Last Few Lectures: Using classes

- Import the name of the class you want:
  import java.math.BigInteger;
- Declare variables using the class name:
  BigInteger salary;
- Create values (objects) of the new type/class by using constructors:
  salary = new BigInteger("111222333");
- Operate on the objects using methods:
  totalCompensation = salary.add(bonus);

Using HelloWorld

- Import the name of the class you want:
  Not needed if .class file in same directory
- Declare variables using the class name:
  HelloWorld foo;
- Create values (objects) of the new type/class by using constructors:
  foo = new HelloWorld();
- Operate on the objects using methods:
  foo.main(null); // Don't worry about null
Using HelloWorld

- Import the name of the class you want:
  Not needed if .class file in same directory
- Declare variables using the class name:
  HelloWorld foo;
- Create values (objects) of the new type/class
  by using constructors:
  foo = new HelloWorld();
- Operate on the objects using methods:
  HelloWorld.main(null);  // main is static

Whoa... what's going on?

Sequential Execution

- Suppose you have a list of tasks:
  - Get money from bank machine.
  - Pick up clothes from dry cleaner.
  - Get lunch from take-out place.

Change of Perspective

- Suppose you have a list of tasks:
  - Get money from bank machine.
  - Pick up clothes from dry cleaner.
  - Get lunch from take-out place.
  - Bank Machine:
    - Check PIN
    - Check balance
    - Dispense cash
  - Dry Cleaner:
    - Find clothes
    - Deliver clothes
  - Restaurant:
    - Take order
    - Prepare food

Change of Perspective

- Suppose you have a list of tasks:
  - Get money from bank machine.
  - Pick up clothes from dry cleaner.
  - Get lunch from take-out place.
  - Bank Machine:
    - Check PIN
    - Check balance
    - Dispense cash
  - Dry Cleaner:
    - Find clothes
    - Deliver clothes
  - Restaurant:
    - Take order
    - Prepare food
Suppose you have a list of tasks:
- Get money from bank machine.
- Pick up clothes from dry cleaner.
- Get lunch from take-out place.

Suppose you have a list of tasks:
- Bank Machine:
  - Check PIN
  - Check balance
  - Dispense cash
- Dry Cleaner:
  - Find clothes
  - Deliver clothes
- Restaurant:
  - Take order
  - Prepare food

Suppose you have some Java statements:
```java
bonus = options.multiply(gain);
total = salary.add(bonus);
System.out.println(total);
```

Suppose you have some Java statements:
```java
bonus = options.multiply(gain);
total = salary.add(bonus);
System.out.println(total);
```

Suppose you have some Java statements:
```java
bonus = options.multiply(gain);
total = salary.add(bonus);
System.out.println(total);
```

Suppose you have some Java statements:
```java
bonus = options.multiply(gain);
total = salary.add(bonus);
System.out.println(total);
```
Change of Perspective

Suppose you some Java statements:

```java
bonus = options.multiply(gain);
total = salary.add(bonus);
System.out.println(total);
```

BigInteger Class:

```java
BigInteger multiply (...) { …
...

BigInteger add (...) { …
...

BigInteger add (...) { …
...

System Class: …
```

How to Create a Class

Client Programmer:

```java
MyClass a;
public class MyClass {
    a.sayHello();
}
```

File MyClass.java

```java
public class MyClass {
    public void sayHello() {
        System.out.println("Hello");
    }
}
```

Questions?