CMSC 131

Object-Oriented Programming I

Scanner, Conditionals, Logical Op

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

- Scanner
- Programming errors
- Debugging
- Conditionals
- Logical Operators

User Input in Java

- We've done output (System.out); what about input?
- Java 5.0 includes the **Scanner class** feature
  - Can use Scanner to create “scanner objects”
  - Scanner objects convert user input into data
- To use Scannner need to *import* a library:

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

Scanner Class Details

- To create a scanner object:
  new Scanner(*input_source*);
  - Input source can be keyboard (System.in), files, etc.
  - Object must be assigned to a variable (e.g. sc)
- Operations
  - nextBoolean()
  - nextByte()
  - nextDouble()
  - nextFloat()
  - nextInt()
  - nextLong()
  - nextShort()

  Returns value of indicated type (reports error if type mismatch)

  - next() Returns sequence of characters up to next whitespace (space, carriage return, tab, etc.)

  - nextLine() Returns sequence of characters up to next carriage return
- **Example:** Scannerl1.java, Scanner2.java

Objects

- About Scanner
  - `Scanner` is a class defined in `java.util.Scanner`
  - `System.in` is a predefined *object* for keyboard input
  - **`sc = new`** `Scanner(System.`*`in`*`)` creates a new *object* in the `Scanner` class and assigns it to `sc`
- Object?
  - A bundle of data (*instance variables*) and operations (*methods*)
  - A class defines both instance variables and methods for objects
  - A class is also a type for objects
  - **`new`** creates new objects in the given class
- We will learn (much) more about objects later

Programming Errors

- Syntax error
  - Violates language's grammar
  - Compiler will catch them
  - Eclipse makes red squiggles under your code
- Run-time errors (exceptions in Java)
  - Something unexpected happens during program execution (e.g., dividing by 0, file not found)
- Semantical/logical errors
  - Program does not generates expected results
  - Can be challenging to uncover

Debugging

- Process of finding and fixing problems
- Important rule while writing programs and to avoid debugging: Incremental Code Development
- Incremental code development
  - Write a little bit, test thoroughly and continue
- Suggestions about writing computer programs can be found on the class web page (Resources section under “Suggestions for Writing Computer Programs”)
- A document about debugging essentials can be found on the class web page (Resources section under “Learning the Essentials of Debugging)

Control Flow and Conditionals

- Control flow → the order in which statements are executed
  - General rule → top to bottom
  - Several Control Structures that change that
- *Conditional statements* → permit control flow to be dependent on (true/false) conditions
  - if
  - if-else

if and if-else

The if and if-else statements have the following forms:

- if (*condition*) {
  statements;
  }
  - tests the condition
  - if true statements are done; otherwise they are skipped
- if (*condition*) {
  statements1;
  } else {
  statements2;
  }
  - tests the condition
  - if true, statements1 is done; otherwise statements2 is done
- **Example:** SimpleIf.java, SimpleIfElse.java

Logical Operators

Used for forming more complex conditions.

- "and" &&

```
if ( temp >= 97 && temp <= 99 ) {
    System.out.println( "Patient is healthy" );
}
```

- "or" ||

```
if ( months >= 3 || miles >= 3000 ) {
    System.out.println( "Change your oil" );
}
```

- "not": !

```
if ( ! phone.equals( "301-555-1212" ) ) {
    System.out.println( "Sorry, wrong number" );
}
```

- Example: LogicalOps1.java, LogicalOps2.java