CSSE 220 Day 7

More Decisions
Iteration
Debugging

Check out *DecisionsAndIteration* from SVN

Questions?

Today: More small topics

- Calculating powers
- Selection operator, `? :`
- `switch` and enumerations
- test coverage
- loops
- debugging

But, first …

- Which is better (if we ignore the fact that the first one is easier to type)?
  - x * x * x
  - Math.pow(x, 3);
- Think about how each would be calculated

`static double`	`pow(double a, double b)` Returns the value of the first argument raised to the power of the second argument.

Statements *vs.* Expressions

- Statements: used only for their *side effects*
  - Changes they make to stored values, control flow,
  - Or input/output (screen, keyboard, files, network, etc)
- Expressions: calculate values

- Many statements contain expressions:

```
if (amount <= balance) {
  balance = balance - amount;
} else {
  balance = balance - OVERDRAFT_FEE;
}
```

Selection Operator

- Lets us choose between two possible values for an expression
- Example:
  - ```
    balance = balance -
      (amount <= balance) ?
                        amount : OVERDRAFT_FEE;
    ```
- Also called the “ternary” operator (Why?)
    ```

Switch Statements: Choosing Between Several Alternatives

```
char grade = …
int points;
switch (grade) {
case ‘A’:
   points = 95;
   break;
case ‘B’:
   points = 85;
   break;
…
default:
   points = 0;
}
```

Can switch on integer, character, or “enumerated constant”

Don’t forget the breaks!

Q2

Example: Convert a Decimal Digit into a Roman Numeral

```
public static String romanDigit(int digit) {
  // return one decimal digit in Roman.
  String result = "";
  switch (digit) {
    case 9:
      result += "IX";
      break;
    case 4:
      result += "IV";
      break;
    case 5: case 6: case 7: case 8:
      result += "V";      // Notice that there is no break in
      digit = digit - 5; // this case. Is this correct?
    default:
      for (int i = 1; i <= digit; i++)
        result += "I";
  }
  return result;
}
```

Enumerated Constants

- Let us specify named sets of values:

```
public enum Suit {
    CLUBS,
    SPADES,
    DIAMONDS,
    HEARTS
}
```

- Then **switch** on them:

```
public String colorOf(Suit s) {
  switch (s) {
    case CLUBS:
    case SPADES:
      return “black”;
    default:
      return “red”;
  }
}
```

Another Enumeration Example

```
public class TryEnums {

        public enum Day {
                SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY,
                 FRIDAY, SATURDAY
        }

        public static void main(String[] args) {
                Day d = Day.SUNDAY;
                System.out.println(d);
                System.out.println(Day.MONDAY.ordinal());
                for (Day d2 : Day.values())
                        System.out.print(d2 + " ");
                System.out.println();
        }
}
```

Output:

```
SUNDAY
1
SUNDAY MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY
```

Boolean Essentials—Like C

- Comparison operators: `<`, `<=`, `>`, `>=`, `!=`, `==`
- Comparing objects: `equals()`, `compareTo()`
- Boolean operators:
  - and: `&&`
  - or: `||`
  - not: `!`

Predicate Methods

- A common pattern in Java:

```
public boolean isFoo() {
    … // return true or false depending on
      // the Foo-ness of this object
}
```

- We tested and implemented isWholeNumber in the Fraction class

Test Coverage

- *Black box testing*: testing without regard to internal structure of program
  - For example, user testing
- *White box testing*: writing tests based on knowledge of how code is implemented
  - For example, unit testing
- *Test coverage*: the percentage of the source code executed by all the tests taken together
  - Want high test coverage
  - Low test coverage can happen when we miss branches of **switch** or **if** statements

Q5

Loops

- while
  for

Java While Loop

- While loop syntax:
  - `while (condition)`
    `  statement`
  - The statement may be a compound statements (i.e. a group of statements enclosed in curly braces.
  - **Caution**: the statement is also allowed to be empty. What happens here?
    - `while (n < 300);`
      `  n*=2;`
  - Can use **break** or **continue** inside any Java loop.

Java For Loop

- For loop syntax:
  - **for** (*initialization* **;** *condition* **;** *update*)
    statement
  - *Once again, the statement is allowed to be compound or empty.*

Sentinel Values: A Loop and a Half

- *Sentinel value*—a special input value not part of the data, used to indicate end of data set
  - `Enter a quiz score, or Q to quit:`

- *A loop and a half*—a loop where the test for termination comes in the **middle** of the loop

- Examples…

Q6,7

Two Loop-and-a-half Patterns

```
// Pattern 1

boolean done = false;
while (!done) {
    // do some work
    if (condition) {
        done = true;
    } else {
        // do more work
    }
}
```

```
// Pattern 2

while (true) {
    // do some work
    if (condition) {
        break;
    }
    // do more work
}
```

Q8

Debugging—Key Concepts

- Breakpoint
- Single stepping
- Inspecting variables

Debugging—Demo

- Debugging Java programs in Eclipse:
  - Launch using the debugger
  - Setting a breakpoint
  - Single stepping: *step over* and *step into*
  - Inspecting variables
- Complete WhackABug exercise,
  then begin HW7 (Pascal Christmas Tree)