Lec 6: Switch and Loops

GUEST LECTURER – JUSTIN HSIA
UNIVERSITY OF CALIFORNIA, SAN DIEGO
CSE 11 - INTRODUCTION TO COMPUTER SCIENCE AND OBJECT-ORIENTED PROGRAMMING: JAVA
Lecture Outline

Control Flow Statements

switch
while loops
do-while loops
break and continue
Switch Statement

Specialized version of if else-if

- All conditionals are equality checks

```java
switch(var) {
    case 3:
        // code3
        break;
    case 8:
        // code8
        break;
    default:
        // code
        break;
}
```

```java
if (var==3) {
    // code3
} else if (var==8) {
    // code8
} else {
    // code
}
```
**switch Fall Through**

Starts executing code when **case** is found and doesn’t stop until it sees **break**

```java
int var = 3;
switch(var) {
    case 1:   out=1;
    case 3:   out=3;
    case 5:   out=5;
    case 7:   out=7;
    case 9:   out=9;
    default:  out=0;
}
```

*out always 0 regardless of var!*
switch Example

Report parity of digit:

```java
switch(digit) {
    case 1:
    case 3:
    case 5:
    case 7:
    case 9: out = "odd"; break;
    case 0:
    case 2:
    case 4:
    case 6:
    case 8: out = "even";
    default: out = "error";
}```
Let’s Draw a Square

Java method signature:

\[
\text{graph.drawRect}(x, y, width, height);
\]

Example:

\[
\text{graph.drawRect}(10, 10, 50, 50);
\]
Let’s Draw a Grid

3x3 for Tic-Tac-Toe:

graph.drawRect(10, 10, 50, 50);
graph.drawRect(60, 10, 50, 50);
graph.drawRect(110, 10, 50, 50);
graph.drawRect(10, 60, 50, 50);
graph.drawRect(60, 60, 50, 50);
graph.drawRect(110, 60, 50, 50);
graph.drawRect(10, 110, 50, 50);
graph.drawRect(60, 110, 50, 50);
graph.drawRect(110, 110, 50, 50);

Battleship (10x10)? Go (18x18)?
Why Loop?

Want to do the same or similar things over and over again

- Save time typing out all instructions

Examples:

- Calculate course grade for all students in class
- Modify all pixels in an image
Loops

There are three looping statements in Java:

- Indefinite – # of repeats not known in advance
  1. while
  2. do-while
- Definite – executes a known # of times
  3. for
**while Loops**

**Syntax:**

```java
while(condition) {
  // code
  // update condition variable
}
```

**Flow chart:**

1. Start
2. **Condition?**
   - **True** → Loop Body
   - **False** → End

---

1/15/2016
CSE 11 WINTER 2016 - LEC 6
while Example 1

```java
int num = 3;
int index = 0;
while ( index < num ) {
    System.out.print( index + " ");
    index++;
}
```

Output:
- 0 1 2

How many times did the loop execute?
- 3
while Example 2

Want to find smallest \textit{factor} > 1 of a number
\begin{itemize}
    \item Factor: natural number divisor
    \item 1 is a factor of all natural numbers – not interesting
\end{itemize}

Example: 30
\begin{itemize}
    \item Factors: 1, 2, 3, 5, 6, 10, 15, 30
\end{itemize}

How can we tell if \texttt{i} is a factor of \texttt{n}?
\begin{itemize}
    \item Modulo! Factor if \texttt{i \% n == 0}
\end{itemize}
// Find a number’s smallest factor > 1
Scanner console = new Scanner(System.in);
System.out.print("Type a number: ");
int number = console.nextInt();
int factor = 2;
while (number%factor != 0) {
    factor++;
}
System.out.println("First factor: "+ factor);

How many times does the loop execute if:

- number = 5  3 times
- number = 4  0 times
- number = 1  ∞ times!
Infinite Loops

These are almost always bad
- Loop body is continuously executing
- Notable exception: embedded systems

Symptom: program never responds
- Force quit or kill process

Common causes
- Bad/missing update code
- Unexpected input/initialization
do-while Loops

Syntax:

do {
    // code
    // update condition variable
} while(condition);

Flow chart:
What does this code do?

Scanner scnr = new Scanner(System.in);
int entered = 0;
do {
    entered++;
    System.out.print("Enter \"quit\" to exit: ");
} while (!scnr.next().equals("quit"));

(A) Runs loop once and then exits
(B) Counts the # of user responses until seeing “quit” (excluding quit)
(C) Counts the # of user responses until seeing “quit” (including quit)
(D) The loop never executes
(E) None of the above
True or False?

I can write any **do-while loop as a while loop**.
- **True**, copy loop body above

I can write any **while loop as a do-while loop**.
- **False**, case where loop body never executed

I can write a definite loop using **while**.
- **True**, use counter variable
Common Looping Errors

Family Feud-style, can you name some of the most common errors with do-while looping?

- Forgetting to update (infinite loop)
- Off-by-one error
- Forgotten semi-colon
- Bad/lack of initialization
Manipulating Body Execution

What if we don’t want the loop body to execute every time?

- `break` – exit loop early
- `continue` – skip rest of this iteration
break Example 1

We saw `break` earlier!

```
switch(variable) {
    case 1:
        //code
        break;
    case 2:
        //code
        break;
    default:
        //code
        break;
}
```
break Example 2

Now in a loop:

```java
int j=0;
while(j < 5) {
    if (j==3) {
        break;
    }
    System.out.print(j + " ");
    j++;
}
```

Prints out: 0 1 2
What’s the output if we use `continue` instead?

```java
int j=0;
while(j < 5) {
    if (j==3) {
        continue;
    }
    System.out.print(j + " ");
    j++;
}
```

(A) 0 1 2 3 3 3...
(B) 0 1 2 3
(C) 0 1 2 3 3 3...
(D) 0 1 2 4
(E) 0 1 2 4 5
Comparison

**break** ends execution early
- Good for “stop” or “kill” commands
- Similar to exit condition, doesn’t make sense to have at top or bottom of loop code

**continue** moves to next comparison
- Good for “skip” commands
- Make sure doesn’t skip over condition variable update
Example: Skip every third num

Do NOT print multiple of 3:

```java
int j = 0;
while(j < 100) {
    if( j%3 == 0 ) {
        continue;
    }
    System.out.println(j);
}
```
Example: Skip every third num

Can we accomplish without using break or continue?

```java
int j = 0;
while(j < 100) {
    if( j%3 != 0 ) {
        System.out.println(j);
    }
}
```

Choice of coding style; lots of equivalences!
Have a great MLK weekend and sleep well!

while ( myself.stillAwake() ) {
    sheep++;
}

©2011 BLOGOGRAPHY.COM