Lecture 6: Loops

GUEST LECTURER – JUSTIN HSIA

UNIVERSITY OF CALIFORNIA, SAN DIEGO

CSE 11 - INTRODUCTION TO COMPUTER SCIENCE AND OBJECT-ORIENTED PROGRAMMING: JAVA
How would you handle the case for when a user selects an option?
Select one of the following:

1. User tasks
2. Office tasks
3. General system tasks
4. Files, libraries, and folders
5. Programming
6. Communications
7. Define or change the system
8. Problem handling
9. Display a menu
10. Information Assistant options
11. iSeries Access tasks

90. Sign off

Selection or command

F3=Exit F4=Prompt F9=Retrieve F12=Cancel F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2005.
Switch Statements

switch(variable) {
    case 3:
        //code
        break;
    case 8:
        //code
        break;
    default:
        //code
        break;
}

### Switch Statements

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

```java
if (variable == 3) {
    //code
}
else if (variable == 8) {
    //code
}
else {
    //code
}
```
What’s the final value of z if program starts with z=0? z=2?

```
switch(z) {
    case 1:
        z*=2;
    case 2:
        z*=3;
    case 5:
        z+=1;
    case 6:
        z--;
    default:
        z=0;
}
```

A) 6,0  B) 0,0  C) 0,6  D) C.E.  E) Runtime error
- If Asked to draw this image. How would you do it with what you know so far.
If Asked to draw this image. How would you do it with what you know so far.

draw(lemon);
draw(lemon);
newLine();
draw(blueThing);
draw(blueThing);
• Check if you can eliminate any candy.

```java
if(square1_1 == square1_2 ||
    square1_1 == square2_1 ||
    square2_1 == square2_2 ||
    square1_2 == sqaure2_2) {
    removeCandy();
    addNewCandy();
    printInspiringMessage();
}
```
Motivating Example

Code instruction-by-instruction?
Drawing example?
Regular polygon example?
Batch example?
Sum/accumulator example?
Why Loop?

Want to do the same thing over and over again
  ◦ Save time typing out all instructions
  ◦ Generalize procedure if changing by same amount
Lecture Outline

Loops

while loops

do-while loops

break and continue
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:
while Example 1

```java
int num = 9;
int indexLemons = 0;
while (indexLemons < num) {
    System.out.print(indexLemons + " ");
    indexLemons++;
}
```
while Example 1

```java
int num = 9;
int indexLemons = 0;
while (indexLemons < num) {
    System.out.print(indexLemons + " ");
    indexLemons++;
}
```

Output:

- 0 1 2 3 4 5 6 7 8

How many times did the loop execute?

- 9
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}
while Example 2

// 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!
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
Stretch Break

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

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

while (nmoves > 0) {
    user_input;
    if(numCherries == 2 &&
        numPeaches == 1) {
        break;
    }
}

break Example 1

We’ve seen break before!

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
**continue Example**

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 4 5
(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

Can we accomplish without using break or continue?