Loops

GEEN163 Introduction to Computer Programming

"Simplicity is prerequisite for reliability."

Edsger W. Dijkstra

TuringsCraft Assignment

- Answer at least 34 of the 62 questions in sections 5.1 – 5.5 of the TuringsCraft tutoring system
- You will earn 3 points for each correct answer
- Due midnight today

Homework

- The GUI number guessing game programming assignment has been posted on Blackboard
- Homework is due Friday at midnight

One or the Other

- With an if – else statement, either the if part or the else part are executed, but never both

```
if ( logical expression ) {
        Executed only if true
} else {
        Executed only if false
}
```

One or maybe the Other

- With an else if statement, second if is evaluated only if the first if is false

```
if ( logical 1 ) {
        Executed only if true
}else if ( logical 2 ){
        Executed only if logical 1 is false
                and logical 2 is true
}
```

nested if

```
if (cat == 3)
    if (bull == 7)
        x = 1;  // cat is 3 and bull is 7
    else
        x = 2;  // cat is 3 and bull is not 7
else
    x = 3;  // cat is not 3, bull doesn't matter
```

Connecting **else** to **if**

- An else statement is always related to the if of the previous block or statement

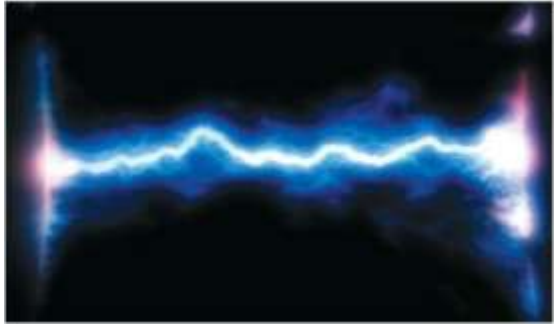
```
if (cow == 3)
   if (bat == 7)
      dog = 1; // cow is 3 and bat is 7
   else
      dog = 3; // cow is not 3 and bat is not 7
```

What is displayed?

```
int rabbit = 3, bunny = 5, hare = 7, pika = 9;
if (rabbit < 4)
        if (bunny < hare)
                pika = 2;
        else
                pika = 6;
System.out.println( pika );
```

A. 2
B. 4
C. 6
D. 9
E. none of the above

Short Circuit Evaluation



- For a logical AND (**&&**) to be true, both sides of the **&&** have to be true
- When evaluating a logical expression with an **&&**, if Java determines that the left side is false, there is no need to evaluate the right side
- Avoiding evaluating the right part of an **&&** can be beneficial if an error might occur

No Need to Continue

- What is the result of the **`if`** ?

```
int hamster= 5, cat = 3;
if (hamster == 47 && [redacted])
```

- Once Java determines the left side of the && is false, the right side does not matter

Potential Divide Problem

```
int gerbil, rat = 17;
System.out.print("Enter a number >");
gerbil = keyboard.nextInt();
if (rat/gerbil < 4) {
        rat = 10;
}
```

- If a zero is read into gerbil, you will get a divide fault if rat/gerbil is executed

Short Circuit Advantage

```
int gerbil, rat = 17;
System.out.print("Enter a number >");
gerbil = keyboard.nextInt();
if (gerbil != 0  &&  rat/gerbil < 4) {
        rat = 10;
}
```

- If a zero is read then gerbil != 0 will be false and rat will not be divided by gerbil

Short Circuit OR

- If either side of a logical OR (||) is true, then the whole expression with be true
- Java stops evaluating an expression with an II once it finds a true part

Short Circuit Advantage

```
int gerbil, rat = 17;
System.out.print("Enter a number >");
gerbil = keyboard.nextInt();
if (gerbil == 0  ||  rat/gerbil < 4) {
        rat = 10;
}
```

- If a zero is read then gerbil == 0 will be true and rat will not be divided by gerbil

Conditional Assignment

- Java provides a *little known* method for putting an if statement in the middle of an expression

  logical expression **?** true part **:** false part

  dog = cat == 0 ? cow : goat;

- if cat is equal to zero, set dog to the value of cow, else set dog to the value of goat.

Conditional Example

```
int cow= 3, cat = 5, dog = 7, goat = 17;
dog = keyboard.nextInt();

cow = (dog == 0 ? cat : goat) + 1;
```

is the same as

```
if (dog == 0 )
        cow = cat + 1;
else
        cow = goat + 1;
```

Clicking Process

1. Click by yourself without discussion
2. Check the results
3. Discuss the question with your team
4. Click as a team

What is displayed?

```
int wren = 2, robin = 5, hawk = 17;
hawk = robin > 4 ? wren-1 : robin+ 1;
System.out.println( hawk );
```

A. 1
B. 2
C. 5
D. 6
E. none of the above

Repeating

- Many programs do the same task many times
- Java provides several ways of creating a program loop
  - while
  - do while
  - for

Java **`while`** statement

- A **`while`** statement is like an **`if`** statement that repeats until the logical expression is false.

```
int cat = 47, sum = 0;
while (cat > 0) {
    cat = keyboard.nextInt();
    sum = sum + cat;
}
```

Loop Parts

- A while loop has a loop condition and a body.
- The body is repeated while the loop condition is true.

```
while (loop condition) {
    // loop body
}
```

`while` loop operation

- When the program execution encounters a while statement, it will check the loop condition
- If the loop condition is false, the loop body will not be executed. Execution will continue with the statements after the loop body
- The body of the loop will be executed if the loop condition is true
- At the end of the body of the loop, the loop condition will be evaluated again. If it is true, the body of the loop will be executed again.

{ Brackets }

- A **`while`** loop can be written without curly brackets around the loop body
- Without brackets, the loop body will be the one statement following the loop condition
- It is recommended that you always use brackets

Example Program

- This is a program to display the average of a series of numbers. The end of the list of numbers is denoted by a value of zero.
- You calculate the average by adding all the values and then dividing by the number of values

```
/*  Example loop program */
import java.util.Scanner;
public class LoopExample {
  public static void main(String[] junk) {
    double num, avg, sum = 0.0;
    int count = 0;
    Scanner keyboard = new Scanner(System.in);
    num = keyboard.nextDouble();
    while (num != 0.0) {
       sum = sum + num;
       count = count + 1;
       num = keyboard.nextDouble();
    }
    avg = sum / count;
    System.out.println("The avg is " + avg);
  }
}
```

Observations about the Program

- The program continues until a zero is read. The while loop checks for a zero
- We need to both sum the numbers and count how many numbers
- There was a “priming” read before the while and another read in the while loop body
- You should not divide until after the loop

Maybe Never

- The logical expression of a while loop is tested for the first time before the statement is executed
- If the logical expression is false the first time, the loop is never executed

```
int hen = 5;
while ( hen < 4 ) {
    hen = hen * hen;
}
System.out.println( hen );
```

What is the final value of dog?

```
int dog = 2, cat = 0;
while ( cat < 3 )  {
   dog = dog + cat;
   cat = cat + 1;
}
```

A. 2

B. 4

C. 5

D. 8

E. none of the above

Infinite Loops

- ***Many*** students have written programs with infinite loops. These programs have a loop that will repeat forever.
- A program with an infinite loop will appear "hung" or "stuck"
- It is necessary that some statement inside a **`while`** loop modifies at least one of the variables used in the **`while`** statement's logical expression.

Infinite Examples

```
int cat = 5, bird = 7;
while (bird > cat) {
    cow = bird + cat;
}

while (bird > cat) {
    bird = bird + cat;
}
```

Is this your clicker?

8EE6BDD5
8EE6C5AD
8EE81A7C
8EF3710C
8F0C0281
8F29EC4A
91FF452B
926701F4
932A5DE4
934328F8

Is this your clicker?

8F093EB8

8F229B36

Looping a Specified Number of Times

- Frequently you may want your program to loop n times.

```
int i = 1, n = 10;
double principle= 10000.0;
while (i <= n) {
        principle = principle * 1.05;
        i++;
}
System.out.println("The value is "+ principle);
```

Looping a Specified Number of Times

- Computer scientist often count starting at zero.

```
int i = 0, n = 10;
double principle= 10000.0;
while (i < n) {
        principle = principle * 1.05;
        i++;
}
System.out.println("The value is "+ principle);
```

What is displayed?

```
int dog = 2, cat = 0;
while ( dog > 0 )  {
   dog = dog + cat;
   cat = cat + 1;
}
System.out.println(dog);
```

A. 2

B. 4

C. 5

D. 8

E. none of the above

Caution

The following loop is wrong:

```
int i=0;
while (i < 10);
{
      System.out.println("i is " + i);
      i = i + 1;
}
```

Practice

- Write a program to compute the future value of an investment. Each year the amount is (1+interest rate) * (previous amount).
- Print the amount for ten years.

Interest Program

```
double amount, interest;
int year=0;
Scanner keyboard = new Scanner(System.in);
amount = keyboard.nextDouble();
interest = keyboard.nextDouble() / 100.0;

        Write a while loop here


}
```

Possible Solution

```
double amount, interest;
int year=0;
Scanner keyboard = new Scanner(System.in);
amount = keyboard.nextDouble();
interest = keyboard.nextDouble() / 100.0;
while (year < 10) {
        amount = amount*(1.0+interest);
         year = year + 1;
        System.out.println(year+" "+amount);
}
```

TuringsCraft Assignment

- Answer at least 34 of the 62 questions in sections 5.1 – 5.5 of the TuringsCraft tutoring system
- You will earn 3 points for each correct answer
- Due midnight today

Homework

- The GUI number guessing game programming assignment has been posted on Blackboard
- Homework is due Friday at midnight