CSCI 1103: Conditionals

Chris Kauffman

Last Updated:
Fri Sep 22 16:33:31 CDT 2017

Logistics

Reading

Eck Ch 3 on conditionals (`if/else`)

Goals

- `if/else`
- Conditional Execution

Project 1

- Due end of weekend
- 2 short programs
- Questions?

Conditionals

- Can get some dynamic behavior in programs with input
- So far CANNOT react to that input
- Following program means to illustrate that

```
> javac StudentDiscount.java

> java StudentDiscount
Current bill is $50.00
Are you a student? (true/false)
true
Applying a 10% discount
Current bill is $45.00

> java StudentDiscount
Current bill is $50.00
Are you a student? (true/false)
false
Sucks being an adult, doesn't it?
Current bill is $50.00
```

Student Discount Program: using `if/else`

```
// Get user input to determine whether discounts apply
public class StudentDiscount{
  public static void main(String args[]){
    double bill = 50.00;
    System.out.printf("Current bill is $%.2f\n",bill);

    System.out.println("Are you a student? (true/false)");
    boolean student = TextIO.getBoolean();
    double discount = 0.00;       // Fraction discount

    // CONDITIONALS
    if(student){
      System.out.println("Applying a 10% discount");
      discount = discount + 0.10; // Increase discount by 10%
    }
    else{
      System.out.println("Sucks being an adult, doesn't it?");
    }

    bill = bill * (1.0-discount);
    System.out.printf("Current bill is $%.2f\n",bill);
  }
}
```

Basic if and if/else Syntax

if only

```
always do this stuff;
and do this stuff;

if(some boolean value is true){
  do this only when true;
  also do this when true;
  ...;
}

always do this stuff;
and this;
```

if/else

```
always do this stuff;
and do this stuff;

if(some boolean value is true){
  do this only when true;
  also do this when true;
  ...;
}
else{
  do this only when false;
  and this when false;
  ...;
}

always do this stuff;
and this;
```

Exercise: Trace Conditional Execution

```
 1  public class SimpleConditions{
 2    public static void main(String args[]){
 3      int sum = 0, a=10, b=15, c=25;
 4      boolean addA = false;
 5      boolean addB = true;
 6      boolean addC = false;
 7
 8      if(addA){
 9        sum = sum + a;
10      }
11      if(addB){
12        sum = sum + b;
13      }
14      if(addC){
15        sum = sum + c;
16      }
17      else{
18        sum = sum * 2;
19      }
20
21      System.out.println("sum is "+sum);
22    }
23  }
```

- Show the output of this program
- Careful of the behavior of the final `if/else`

Exercise: Trickier Conditional Execution

```
 1  public class TrickyConditions{
 2    public static void main(String args[]){
 3      int sum = 0, a=5, b=7;
 4      boolean addA = true;
 5      boolean addB = false;
 6      boolean div2 = true;
 7
 8      if(addA){
 9        sum = sum + a;
10        addB = true;
11        div2 = false;
12      }
13      if(addB){
14        sum = sum + b;
15      }
16      else{
17        sum = sum - b;
18      }
19      if(div2){
20        sum = sum / 2;
21        addA = false;
22      }
23      else{
24        sum = sum*2;
25      }
26      System.out.printf("sum: %d\n",sum);
27      System.out.printf("addA: %s\n",addA);
28      System.out.printf("addB: %s\n",addB);
29      System.out.printf("div2: %s\n",div2);
30    }
31  }
```

- Show the output of this program
- Note the values of `a,b,c` change along with sum
- Show the PATH through the program (line numbers executed)

Answer: Trickier Conditional Execution (1)

```
 1  public class TrickyConditions{
 2    public static void main(String args[]){
 3      int sum = 0, a=5, b=7;
 4      boolean addA = true;
 5      boolean addB = false;
 6      boolean div2 = true;
 7
 8      if(addA){
 9        sum = sum + a;
10        addB = true;
11        div2 = false;
12      }
13      if(addB){
14        sum = sum + b;
15      }
16      else{
17        sum = sum - b;
18      }
19      if(div2){
20        sum = sum / 2;
21        addA = false;
22      }
23      else{
24        sum = sum*2;
25      }
26      System.out.printf("sum: %d\n",sum);
27      System.out.printf("addA: %s\n",addA);
28      System.out.printf("addB: %s\n",addB);
29      System.out.printf("div2: %s\n",div2);
30    }
31  }
```

1st conditional: Consequence

CPU: Line 8
MEMORY:

Box	Value
sum	0
a	5
b	7
addA	true
addB	false
div2	true

CPU: Line 9
MEMORY:

Box	Value
sum	0
a	5
b	7
addA	true
addB	false
div2	true

CPU: Line 12
MEMORY:

Box	Value
sum	5
a	5
b	7
addA	true
addB	true
div2	false

Notice `addB` and `div2` changed

Answer: Trickier Conditional Execution (2)

```
 1  public class TrickyConditions{
 2    public static void main(String args[]){
 3      int sum = 0, a=5, b=7;
 4      boolean addA = true;
 5      boolean addB = false;
 6      boolean div2 = true;
 7
 8      if(addA){
 9        sum = sum + a;
10        addB = true;
11        div2 = false;
12      }
13      if(addB){
14        sum = sum + b;
15      }
16      else{
17        sum = sum - b;
18      }
19      if(div2){
20        sum = sum / 2;
21        addA = false;
22      }
23      else{
24        sum = sum*2;
25      }
26      System.out.printf("sum: %d\n",sum);
27      System.out.printf("addA: %s\n",addA);
28      System.out.printf("addB: %s\n",addB);
29      System.out.printf("div2: %s\n",div2);
30    }
31  }
```

2nd conditional: consequence

Since addB is now `true`, do the `if` part rather than `else` part

CPU: Line 13
MEMORY:

Box	Value
sum	5
a	5
b	7
addA	true
addB	true
div2	false

CPU: Line 14
MEMORY:

Box	Value
sum	5
a	5
b	7
addA	true
addB	true
div2	false

CPU: Line 15
MEMORY:

Box	Value
sum	12
a	5
b	7
addA	true
addB	true
div2	false

Answer: Trickier Conditional Execution (3)

```
 1  public class TrickyConditions{
 2    public static void main(String args[]){
 3      int sum = 0, a=5, b=7;
 4      boolean addA = true;
 5      boolean addB = false;
 6      boolean div2 = true;
 7
 8      if(addA){
 9        sum = sum + a;
10        addB = true;
11        div2 = false;
12      }
13      if(addB){
14        sum = sum + b;
15      }
16      else{
17        sum = sum - b;
18      }
19      if(div2){
20        sum = sum / 2;
21        addA = false;
22      }
23      else{
24        sum = sum*2;
25      }
26      System.out.printf("sum: %d\n",sum);
27      System.out.printf("addA: %s\n",addA);
28      System.out.printf("addB: %s\n",addB);
29      System.out.printf("div2: %s\n",div2);
30    }
31  }
```

Third conditional: alternative

`div2` changed to `false`, go to `else` part

CPU: Line 19
MEMORY:

Box	Value
sum	12
a	5
b	7
addA	true
addB	true
div2	false

CPU: Line 24
MEMORY:

Box	Value
sum	12
a	5
b	7
addA	true
addB	true
div2	false

CPU: Line 25
MEMORY:

Box	Value
sum	24
a	5
b	7
addA	true
addB	true
div2	false

Comparing Numbers

- Often want to compare things to decide what to do
- Java is equipped with standard number comparison operators

```
int x = 10, y = 20;              // Variables

boolean xLTy   = x < y;          // Less than
boolean xLTE15 = x <= 15;        // Less than or equal to

boolean xGTy   = x > y;          // Greater than
boolean xGTE10 = x >= 10;        // Greater than or equal to

boolean xEQy   = x == y;         // Equal to (shallow)
boolean xNEy   = x != y;         // Not equal to (shallow)
```

- It is common to use comparisons directly in `if()`

```
if(x > y){
  System.out.println("x is king");
}
else{
  System.out.println("y might be higher");
}
```

Classic Exercise: Max of 3 Inputs

Spec

- Program prompts for 3 integers, uses `TextIO.getInt()` to retrieve them
- Uses a series of `if()` checks to determine the maximum number
- Prints max number at the end

Start your code...

```
public class Max3{
  public static void main(String args[]){
    System.out.println("Enter 3 integers:");
    int a = TextIO.getInt();
    int b = TextIO.getInt();
    int c = TextIO.getInt();
    // YOUR CODE BELOW
    ...
```

Demos

```
> javac Max3.java

> java Max3
Enter 3 integers:
6 7 1
Max is 7

> java Max3
Enter 3 integers:
12 3 19
Max is 19

> java Max3
Enter 3 integers:
21 18 16
Max is 21
```

Answer: Max of 3 Inputs

```
// Program that determines the maximum of three numbers using
// conditionals
public class Max3{
  public static void main(String args[]){
    System.out.println("Enter 3 integers:");
    int a = TextIO.getInt();
    int b = TextIO.getInt();
    int c = TextIO.getInt();

    int max = a;
    if(b > max){
      max = b;
    }
    if(c > max){
      max = c;
    }
    System.out.println("Max is "+max);
  }
}
```

Combining Conditions

Recall the boolean operators

```
boolean a=true, b=false;
boolean x = a && b; // logical AND: true only if both a,b are true
boolean y = a || b; // logical OR:  false only if both a,b are false
boolean z = !a;     // logical NOT: flips true to false, false to true
```

These are often used to combine numeric checks.

```
int x=1, y=2, z=3;
if( ( (x<y) && ((x+y)!=z) ) || z<4){
  System.out.println("Do I get printed?");
}
```

Operator precedence is roughly: arithmetic, comparison, AND, OR

```
if( x<y && x+y!=z || z<4){   // Equivalent to previous
  System.out.println("Do I get printed?");
}
```

Use parentheses to indicate your intended order to stay sane

Exercise: In Ascending Order

Spec

- Program prompts for 4 integers, uses `TextIO.getInt()` to retrieve them
- Determines if inputs are in order from smallest to largest, ties allowed
- Use conditionals, comparisons, boolean combiners

Start your code...

```
// Enter 4 numbers, check for ascending order
public class Ascending{
  public static void main(String args[]){
    System.out.println("Enter 4 integers:");
    int a = TextIO.getInt();
    int b = TextIO.getInt();
    int c = TextIO.getInt();
    int d = TextIO.getInt();
    // YOUR CODE HERE
    ...
```

Demos

```
> javac Ascending.java

> java Ascending
Enter 4 integers:
1 3 5 9
Numbers are ascending

> java Ascending
Enter 4 integers:
1 3 3 9
Numbers are ascending

> java Ascending
Enter 4 integers:
1 3 9 5
Out of order

> java Ascending
Enter 4 integers:
9 3 5 1
Out of order
```

Answer: In Ascending Order

```
// Enter 4 numbers, check for ascending order
public class Ascending{
  public static void main(String args[]){
    System.out.println("Enter 4 integers:");
    int a = TextIO.getInt();
    int b = TextIO.getInt();
    int c = TextIO.getInt();
    int d = TextIO.getInt();

    if(a<=b && b<=c && c<=d){
      System.out.println("Numbers are ascending");
    }
    else{
      System.out.println("Out of order");
    }
  }
}
```

Variation: Detect order type as one of

- Ascending
- Strictly Ascending (no ties)
- All equal
- Descending
- Strictly Descending (no ties)
- Out of order

Chaining: Exclusive `if / else if / else`

- Common problem: want to select one thing to do based on conditions
- Ex: Discounts, best only. Compare these two

From `BuggyDiscount.java`

```
double discount = 0.0;
if(birthday){
  discount = 0.20;
}
if(student){
  discount = 0.15;
}
if(coupon){
  discount = 0.10;
}
```

From: `ChainedConditions.java`

```
double discount = 0.0;
if(birthday){
  discount = 0.20;
}
else if(student){
  discount = 0.15;
}
else if(coupon){
  discount = 0.10;
}
```

Exercise: Mutual Exclusion and Modulo

- Program `Div8.java`
- Get an integer from the user
- Determine if it is evenly divisible by
  - 8
  - 4
  - 2
  - (sucks) Not divisible by any of these
- Print message only for the biggest of these
- Use a chain of `if/else if/../else`

```
> javac DivIt.java
> java DivIt
Enter an in (ex: 22):
12
12 divisible by 4
> java DivIt
Enter an in (ex: 22):
16
16 divisible by 8
> java DivIt
Enter an in (ex: 22):
6
6 divisible by 2
> java DivIt
Enter an in (ex: 22):
15
15 sucks
> java DivIt
Enter an in (ex: 22):
24
24 divisible by 8
```

Answer: Mutual Exclusion and Modulo

```
// Use mutual exclusion to print whether a given number is divisible
// by 8, 4, 2, but only the largest of these
public class DivIt{
  public static void main(String args[]){
    System.out.println("Enter an in (ex: 22):");
    int num = TextIO.getInt();
    if(num % 8 == 0){
      System.out.printf("%d divisible by 8\n",num);
    }
    else if(num % 4 == 0){
      System.out.printf("%d divisible by 4\n",num);
    }
    else if(num % 2 == 0){
      System.out.printf("%d divisible by 2\n",num);
    }
    else{
      System.out.printf("%d sucks\n",num);
    }
  }
}
```

Nesting Conditionals

- Many programming elements can be nested: placed within another context
- Conditionals can be nested
- Can nest conditionals very deeply, but this makes reading difficult
- 2 levels of nesting is quite common

```
always do this;

if(condition1){
  do some stuff;
  more stuff;
  if(condition2){
    only if condition2 (and condition1);
  }
  else{
    only if NOT condition 2;
  }
}
else{
  only if not condition 1;
}

always do this;
```

Example: NestedDivIt

```
// Use nested if-elses to print whether a given number is divisible
// by 8, 4, 2, but only the largest of these.
// NOTE: this is MUCH harder to read than the chained if/else version
public class NestedDivIt{
  public static void main(String args[]){
    System.out.println("Enter an in (ex: 22):");
    int num = TextIO.getInt();
    if(num % 2 == 0){
      if(num % 4 == 0){
        if(num % 8 == 0){
          System.out.printf("%d divisible by 8\n",num);
        }
        else{
          System.out.printf("%d divisible by 4\n",num);
        }
      }
      else{
        System.out.printf("%d divisible by 2\n",num);
      }
    }
    else{
      System.out.printf("%d sucks\n",num);
    }
  }
}
```

Elegance and Conditionals

- With conditionals, there may be multiple ways to achieve the same behavior
- Chain of exclusive `if/else if/..`, Nesting Conditions, Inversion of conditions

```
boolean a, b; int x=0;
...;
if(a && b){
  x = 3;
}
else if(a){
  x = 2;
}
else if(b){
  x = 1;
}
```

```
boolean a, b; int x=0;
...;
if(a){
  if(b){
    x = 3;
  }
  else{
    x = 2;
  }
}
else if(b){
  x = 1;
}
```

```
boolean a, b; int x=0;
...;
if(a && b){
  x = 3;
}
else if(a && !b){
  x = 2;
}
else if(!a && b){
  x = 1;
}
```

It takes experience to know what to do but the goal is always correctness which is tied up with human understandability

The Other Conditional: `switch()`

```
switch ( N ) {   // (Assume N is an integer variable.)
   case 1:
      System.out.println("The number is 1.");
      break;
   case 2:
   case 4:
   case 8:
      System.out.println("The number is 2, 4, or 8.");
      System.out.println("(That's a power of 2!)");
      break;
   case 3:
   case 6:
   case 9:
      System.out.println("The number is 3, 6, or 9.");
      System.out.println("(That's a multiple of 3!)");
      break;
   case 5:
      System.out.println("The number is 5.");
      break;
   default:
      System.out.println("The number is 7 or is");
      System.out.println("outside the range 1 to 9.");
}
```

- Switch looks cool but is actually stupid
- Hard to use: must `break` or fall through
- Previously only worked for `int`-family, can do `String` more recently
- May enable tiny amount of optimization by compiler at the immense expense of human readability
- Comes in handy once in a blue moon but largely suggest stick to `if/else`