Making Decisions video notes

- Unless we do something to change it, our programs use what we call sequential processing. This means that the Java programs that we write in our programs are executed in a sequence.

- The programs we write always execute in the same order. The data can be different, but the data will be treated in the same way. We want to the program to react differently to different input and to make decisions based on the inputs it has.

Example of a decision in pseudocode:

IF the light is green THEN

Keep going

ELSE

Stop the car

- This is often referred to as selection. We’re selecting which actions to take based on what information we have.

- Java If/Else statement:

if (condition)

{

// Stuff to do when condition is true

}

else

{

// Stuff to do when condition is false

}

- Conditions: evaluate to true of false.

Boolean variable:

if (isMarried)

{

System.out.print(“Enter spouse’s name: “ );

spouseName = keyboard.getNextLine( );

}

else

{

spouseName = “N/A”;

}

- Comparison condition: For this purpose, we have six relational operators.

They don’t have to be numbers but can be characters too.

( = = ) equals operator

( ! = ) not equals

( < ) less than

( > ) greater than

( <= ) less than or equal to

( => ) greater than or equal to

Example:

if (piecesOfCandy % numPeople == 0)

{

System.out.println(“Everyone gets the same amount.”);

}

else

{

System.out.println(“There will be leftovers.”);

}

Example:

if (num1 < num2)

{

smallerValue = num1;

}

else

{

smallerValue = num2;

}

- When we’re comparing values that are floating point numbers (doubles or floats) we must be careful. We can’t count on the accuracy of our floating point numbers, so instead, we will get the difference between the two numbers and see if it is small enough for the two compared numbers to qualify as equal.

- Note that this is a great place to use the absolute value method from the math class.

Example of floating point comparison:

If (Math.abs(num1 - num2) < 0.00001)

- Logical operators are used to combine and negate conditions:

( ! ) The Not operator. Also called a bang. Has a high precedence, so when must applying to a complex condition, we must be careful of our placement with regards to parentheses around the condition.

( && ) The And operator. Make sure to include both ampersands. One ampersand is a valid operator but will not do a logical And. If both sides of the And are true, then the output will be true.

( || ) The Or operator. Or will be true unless both sides are false.

Example conditions:

If ( !done )

“ If not done” Done here must be a Boolean variable. If done is true, not done will be false. If done is false, not done will be true.

If ( letter >= ‘a’ && letter <= ‘z’ )

Is the letter in the range of the lowercase letters? Note that we would get tempted to write this the way we would write it in English. Which might be, a is less than or equal to letter is less than or equal to z. We must have the two separate conditions and the And operator to express this concept.

‘a’ < letter < ‘ z’ WE CANNOT DO THIS IN JAVA.

If (letter = = ‘a’ || letter = = ‘e’ ||

letter = = ‘I’ || letter = = ‘o’ ||

letter = = ‘u’ )

If letter equals ‘a’ or ‘e’ or ‘i’ or ‘o’ or u.’

PRECEDENCE OF LOGICAL OPERATORS:

1) Not

2) And

3) Or

Suppose a Java program has the condition (num%2 == 1 && num > 10). This condition will be true when

1. num is even and num is also bigger than 10.
2. num is odd and num is also bigger than 10.
3. num is even or num is bigger than 10 (or both)
4. num is odd or num is bigger than 10 (or both).

- We don’t always need an “else” when we’re using “if”

Example:

If I’m calculating pay and there’s overtime, I do the same thing for the regular pay either way. If there is overtime pay, I have to do something special. There’s nothing to go in the else clause.

Be careful to leave out the else clause. So if we get tempted to have an empty “if” clause, change the condition and use not if we have to. Set it up so we have just an “if” clause because an empty “else” clause we can get rid of, but an empty “if” is confusing.

If (hoursWorked > 40)

{

// add overtime

}