Control

if, for, while, do while, etc.
if Statement

- You’ve seen it already.
- Here’s the Java syntax.

```java
if(boolean expression)
{
    //when true do this
    //can have as much stuff in here as desired
}
else
{
    //when false do this
    //can have as much stuff in here as desired
}
```
Boolean Expression

Anything that evaluates to true or false.

• i < 10
  • if(i<10)
• height >= 1.1
• x == 2.3
  • if(x == 2.3)
• cashValue != 100.00
• true
  • if(true)
• false
  //boolean value!
• boolean wellWritten = true;
  • if(wellWritten)
  //boolean value!
Compound Boolean Expressions

- Can check two things at once
  - && means “AND”
    - if( (cash > 10.00) && (cash < 20.00) )
    - if(happy && tired)
    - if( (cash % 25 != 0) && haveNoChange)
      
      ```java
      System.out.println("Sorry cannot give you any change.");
      ```

  - || means “OR”
    - if(ecstatic || happy)
    - if( ecstatic || (happy && giddy) )
    - if((date == 28 && isFebruary) || (date == 31 && isJanuary))
Only Use Booleans!

- The following won’t work.

```java
if(i == 3 || 4)
{
    System.out.println (“hah, got in here!”);
}
```

- Why not?
  - Remember your order of operations! Left to right.
  - So starts with “i == 3” and decides if it is true or false.
  - Then goes to “4” and decide if it is true or false.
    - Nonsensical! “4” is not a Boolean! Each part has to be a Boolean expression!

- So must rewrite this way.

```java
if(i == 3 || i == 4)
{
    System.out.println (“hah, got in here!”);
}
```
Methods and Booleans

Remember a method?
- They can return a value.
  - Examples are ".toUpperCase()", ".toLowerCase()" for String
  - These methods return Strings.
    - remember “BUZZ OFF”?

- Some methods return booleans
  - String s = “hello”;
    if(s.startsWith(“h”))
    {
      ...
    }
  if(s.equals(“goodbye”))
  {
    ...
  }

Tricky: to compare Strings must use “.equals”. 
“= =” will give the wrong answer.
String myName = JOptionPane.showInputDialog("Please enter my name.");
String yourName = JOptionPane.showInputDialog("Please enter your name.");
if(!myName.equalsIgnoreCase(yourName))
{
    if(myName.startsWith(yourName.substring(0,1))
    {
        JOptionPane.showMessageDialog(null, "Whoa, same first letter.");
    }
}
else
{
    JOptionPane.showMessageDialog(null, "Whoa, same name dude.");
}
else-if Statements

Does same thing as previous code.

String myName = JOptionPane.showMessageDialog("Please enter my name.");
String yourName = JOptionPane.showMessageDialog("Please enter your name.");
if(myName.equalsIgnoreCase(yourName))
{
    JOptionPane.showMessageDialog(null, "Whoa, same name dude.");
}
else if(myName.startsWith(yourName.charAt(0)))
{
    JOptionPane.showMessageDialog(null, "Whoa, same first letter.");
}
else-if Chains

Can do as many as you want.

```c
if(temp >= 100)
{
    if(humidity > 50) //percent
        {
            ...
        }
    else if(…)
        {
            ...
        }
} else if(temp > 80 && temp < 100)
{
    ...
}
else if(…)
{
    ...
}
else if(…)
{
    ...
}
else if(…)
{
    ...
}
else if(…)
{
    ...
}
```
switch: A Special “if”

Evaluates an expression and then chooses the correct space

```
switch(numberOfDonuts)
{
    case 0:
        System.out.println("Well, that's not good.");
        break;
    case 1:
        System.out.println("Yummy.");
        break;
    case 2:
        System.out.println("Yummy.");
        System.out.println("What's your name?");
        break;
}
```

more than one line allowed

must evaluate to an integer

exits the switch statement

continued next page
switch (continued)

```java
    case 3:
    case 4:
    case 5:
        System.out.println("Too many?");
        break;
    case 10:
        System.out.println("Hi Homer!");
        break;
    default:
        System.out.println("Hi Marge! ");
        break;
}
```

- without a `break`, just goes to the next case
- don’t have to include every case
- default handles any other case
switch(cLetterGrade) {
    case 'A':
        System.out.println("Well done.");
        break;
    case 'Z':
        System.out.println("Your grade makes no sense.");
        break;
    case 'B':
        System.out.println("Well done.");
        break;
    case 'a':
        cLetterGrade = 'A';
        System.out.println("Well done.");
        break;
    }
    "char" act as an "int". So ok!
    cases do not have to be in order
    default not necessary (but is wise)
switch as an “if”

- Can rewrite any switch statement as an else-if!
  - Do it for the previous switch statement (class exercise)

- Here’s the start…

```java
if(cLetterGrade == 'A')
{
    ...
}
else if( ... )
...
```

- How do you include the default?
  - add an “else” at the end.
for Loop

- Just like you’ve already seen.
- Java syntax:

```java
for(int i= 0; i<=10; i++)
{
    ...
}
```
- This means “for i = 0 to 10”
  - The i = 0 initializes the loop.
  - The i<=10 tells the loop to continue as long as this is true.
  - The i++ increments i.
  - The “int i” is a convenience. Could be declared elsewhere.
for Examples

How many times will each of these print “yo”?

```java
for(int numHours = 0; numHours < 25; numHours++)
{
    System.out.println("yo");
}

int numHours;
for(numHours = 1; numHours <= 24; numHours++)
{
    System.out.println("yo");
}
System.out.println(numHours);

for(int i = 1; i <= 24; numHours++)
{
    System.out.println("yo");
}
```

Note block scope for variable numHours. Have to re-declare to use it outside the for loop.

Careful… Infinite loop! But compiles because we declared numHours outside the loop.
Common Mistakes

Five common mistakes. Can you find them?

```java
int sumOfEvens = 0;
for(int i = 10; i>0; i--){
    if(i % 2 == 0){
        int sumOfEvens += i;
    }
}
System.out.println("Sum of evens is "+sumOfEvens);
System.out.println("Last value of i was "+i);
These are ok!
```
while Loops

- Just like you’ve already seen.
- Java syntax:

```java
while(boolean expression)
{
    //whatever you want
}
```

- The boolean expression is the same as used in “if” statements.
//This program keeps going until it finds a number that is not prime. Starts with 1001.
int number = 1001;
boolean prime = true;
while(prime)
{
    for(int i=2; i<number/2; i++)
    {
        if(number % i == 0)
        {
            prime = false;
        }
    }

    if( !prime )
    {
        System.out.println("Number "+number+"is not prime.");
    }

    number++; //checks next integer until it finds a number that isn’t prime.
}
How Many Times Does It Print “yo”?

```java
int n = 0;
while(n < 10) {
    System.out.println("yo");
    n++;
}

while(n < 5) {
    System.out.println("yo");
    n++;
}

int m = 0;
while(m < 5) {
    System.out.println("yo");
}

int i = 0;
while(i >= 23) {
    System.out.println("yo");
    i++;
}

boolean happy = true;
while(happy) {
    System.out.println("yo");
    happy = false;
}
```

- **never** because `n = 10`
- **never** because `n = 10`
- **infinite** – doesn’t increment!
- **once**
- prints once – semicolon causes weirdness
- **never** if get rid of extra semi-colon (happy is false)
**do-while Loop**

- “while” loop checks *before* it starts.
  - May never do the stuff inside.

- “do-while” checks *after* it has gone through once
  - always does the stuff inside at least once!
do-while Syntax

- Java syntax:

```java
do {
    ...this stuff
} while(boolean expression);
```

- Same as while loop, but checks *after* doing the `{…}` block
  - after the “body” of the loop
import java.awt.Toolkit;
import javax.swing.JOptionPane;

public class Alarm {
    public static void main(String[] args) {
        boolean notAwake = true;
        do {
            for (int i = 0; i < 100; i++) {
                Toolkit.getDefaultToolkit().beep();
            }
            String answer = "";
            while (!answer.equals("y") && !answer.equals("n")) {
                answer = JOptionPane.showInputDialog("Are you Awake? Enter y or n.");
                if (answer.equals("y")) {
                    notAwake = false;
                }
            }
        } while (notAwake);
        System.exit(0);
    }
}
Substitute Loops

- Each loop can be used instead of the other.
  - Just choose the most convenient.

- Choose
  - “for” if need fixed number of iterations.
  - “while” if need indefinite number of iterations.
  - “do-while” if indefinite number but need at least once.
Factorial Example With “for”

Calculate n-factorial
  - e.g. 5! = 5 * 4 * 3 * 2 * 1

```java
public static void main(String[] args) {
    int factorial = 1;
    String sNumber = JOptionPane.showInputDialog("Please enter an integer number.");
    int iNumber = Integer.parseInt(sNumber);

    for(int n = 2; n<=iNumber; n++) {
        factorial *= n;
    }

    JOptionPane.showMessageDialog(null, "factorial of \(\text{\textit{+ iNumber +\" is\"}}\) \text{+ factorial +\"}}");

    System.exit(0);
}
```

What needs to be added to make the code complete? e.g.,
import javax.swing.JOptionPane…
public static void main(String[] args) {
    int factorial = 1;

    String sNumber = JOptionPane.showInputDialog("Please enter an integer number.");
    int iNumber = Integer.parseInt(sNumber);

    int n = 2;
    while(n<=iNumber) {
        factorial *= n;
        n++;
    }

    JOptionPane.showMessageDialog(null, "factorial of " + iNumber + " is" + factorial + ".");

    System.exit(0);
}
public static void main(String[] args) {
    int factorial = 1;

    String sNumber = JOptionPane.showInputDialog("Please enter an integer number.");
    int iNumber = Integer.parseInt(sNumber);

    int n = 2;
    do {
        factorial *= n;
        n++;
    } while(n<=iNumber);

    JOptionPane.showMessageDialog(null, "factorial of "+ iNumber +" is"+ factorial +".");

    System.exit(0);
}