Topic 04: Loops
(advanced)

- Equivalency
- More complex for() loops
- More advanced nesting
- do ... while()
- break and continue

Special Topic 4.1
A Template for `for()`

- Choose a type and identifier for your `loop` variable
- Choose `how many times` you want the loop to run
- Write a `loop body`

- Cut and paste:
  ```
  for (int i=0; i<10; i++)
    body;
  ```
for (int i=0; i<10; i++)
    System.out.println("i="+i);

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

These two loops are equivalent.
Can you match up the pieces?
for (int i=0; i<10; i++)
    System.out.println("i="+i);

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

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

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

int i=0;
while (i<10)
{
    System.out.println("i="+i);
    i++;
}
while() as a Flowchart

- **Condition**
  - False
  - True

- **Body**
for() as a Flowchart
Equivalency

• Every `for()` loop can be rewritten as a `while()` loop

• But does it work the other way around?
Unusual `for()` Loops

- Any of the 3 parts of a `for()` loop can be omitted
- But you must leave the semicolons in place

```java
for(; i<10; i++)
for(int i=0; i<100; )
for(;;)
```
Unusual \texttt{for()} Loops

- Any of the 3 parts of a \texttt{for()} loop can be omitted
- But you must leave the semicolons in place

\begin{itemize}
  \item \texttt{for(; i<10; i++)}
  \item \texttt{for(int i=0; i<100; )}
  \item \texttt{for(;;)}
\end{itemize}

Skip the init step.
Unusual \texttt{for()} Loops

- Any of the 3 parts of a \texttt{for()} loop can be omitted
- But you must leave the semicolons in place

\begin{verbatim}
for(; i<10; i++)
for(int i=0; i<100; )
for(;;)
\end{verbatim}

Skip the increment step.
Unusual `for()` Loops

- Any of the 3 parts of a `for()` loop can be omitted
- But you must leave the semicolons in place

```java
for(; i<10; i++)  // Infinite loop!
for(int i=0; i<100; )
for(;;)         // Infinite loop!
```
Unusual `for()` Loops

• Any of the 3 parts of a `for()` loop can be omitted
• But you must leave the semicolons in place

```java
for(; i<10; i++)
for(int i=0; i<100; )
for(;;)  // Infinite loop!
```
while (somethingIsTrue) {
    ...
}

for (; somethingIsTrue;)
{
    ...
}

These two loops are equivalent.
Topic 04: Loops (advanced)

- Equivalency
- More complex for() loops
- More advanced nesting
- do ... while()  
- break and continue  

Special Topic 4.1
More Advanced \texttt{for()} Loops

- You can drop (almost) anything into the three parts of a \texttt{for()} loop:
  - \texttt{Init}: Variable declarations and assignment
  - \texttt{Condition}: Any boolean expression
  - \texttt{Increment}: Any statement (except declaration)
More Advanced \texttt{for()} Loops

\begin{verbatim}
for (int i=10; i>=0; i--)
{
   ...
}
\end{verbatim}

\textbf{In Class:}
What does this loop do?
More Advanced for() Loops

```c++
for (int i=3; ; i+=2)
{
    ... 
}
```

**In Class:**
What does this loop do?
More Advanced \texttt{for() } Loops

\begin{verbatim}
for (int i=1;
i<=1000*1000*1000;
i*=2)
{
    ...
}
\end{verbatim}

\textbf{In Class:}
What does this loop do?
Topic 04: Loops (advanced)

- Equivalency
- More complex for() loops
- More advanced nesting
- do ... while()
- break and continue

Special Topic 4.1
for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        int val = i+j;
        System.out.print(val);
    }
    System.out.println();
}

Output:
???
Addition Table

```java
for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        int val = i+j;
        System.out.print(val);
    }
    System.out.println();
}
```

This is the outer loop. You can think of it like a tiny method.

This “method” runs three times:
i=0, i=1, i=2

The “method” runs a loop, and then calls println().
Addition Table

for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        int val = i+j;
        System.out.print(val);
        System.out.println();
    }
}

This is the inner loop.

The entire loop runs several times.
Addition Table

```
for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        int val = i+j;
        System.out.print(val);
    }
    System.out.println();
}
```

This is the output from this snippet:

```
012
123
234
```

Each line is printed by one pass of the outer loop.
for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        int val = i+j;
        System.out.print(" "+val);
    }
System.out.println();
}
Addition Table

for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        if (j != 0)
            System.out.print(" ");
        int val = i+j;
        System.out.print(val);
    }
    System.out.println();
}
Addition Table

```
for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        if (j != 0)
            System.out.print(" ");
        int val = i+j;
        System.out.print(val);
    }
    System.out.println();
}
```

This puts spaces between the elements, but not at the beginning:

```
0 1 2
1 2 3
2 3 4
```
Spacers

• This is a common idiom!

    Value [space] Value [space] Value

• Print the [space] on every iteration **EXCEPT** the first

    for (int i=0; i<100; i++)
    {
        if (i != 0)
            print(“ [space] ”);
        print(“Value”);
    }
Remember This?

System.out.println("input="+val);
while (val % 2 == 0)
{
    System.out.println("FACTOR: 2");
    val /= 2;
}
System.out.println("leftover="+val);
Prime Factorization

```java
System.out.println("input="+val);

int factor = 2;
while (val > 1)
{
    while (val % factor == 0)
    {
        System.out.println("FACTOR: "+factor);
        val /= factor;
    }
    factor++;
}
```
System.out.println("input=\"+val+\";\n
int factor = 2;\nwhile (val > 1) \n{\n    while (val % factor == 0) \n    {\n        System.out.println("FACTOR: \"+factor+\";\n        val /= factor;\n    }\n    factor++;\n}
Prime Factorization

System.out.println("input="+val);

int factor = 2;
while (val > 1)
{
    while (val % factor == 0)
    {
        System.out.println("FACTOR: "+factor);
        val /= factor;
    }
    factor++;
}

This advanced us to the next factor.
System.out.println("input=":val);

int factor = 2;
while (val > 1)
{
    while (val % factor == 0)
    {
        System.out.println("FACTOR: "+factor);
        val /= factor;
    }
    factor++;
}

This loops until we have pulled out every factor.
Prime Factorization

System.out.println("input=":val);
for (int factor=2; val > 1; factor++)
{
    while (val % factor == 0)
    {
        System.out.println("FACTOR: "+factor);
        val /= factor;
    }
}

This is the same code, just using a for() loop.
Prime Factorization

```java
System.out.println("input="+val);
for (int factor=2; val > 1; factor++)
{
    for (; val % factor == 0; val /= factor)
    {
        System.out.println("FACTOR: "+factor);
    }
}
```

This is the same code, just using two for() loops.
System.out.println("input="+val);

for (int factor=2; val > 1; factor++)
    for (; val % factor == 0; val /= factor)
        System.out.println("FACTOR: "+factor);

This is the same code, just without unnecessary curly braces { } .
Topic 04: Loops (advanced)

- Equivalency
- More complex `for()` loops
- More advanced nesting
- `do ... while()`  4.4
- `break` and `continue`  Special Topic 4.1
do ... while()

- Exactly like `while()`, except that we're guaranteed to run the body at least once

```plaintext
do
{
    body;
}
while (condition);
```
Remember \texttt{while()} ?

\begin{figure}
\centering
\begin{tikzpicture}
  \node [diamond,draw,fill=blue!20] (cond) {condition};
  \node [rectangle,draw,fill=blue!20] (body) at (cond -| body) {body};
  \draw[->] (cond -| false) -- (false) node[above] {False};
  \draw[->] (cond -| true) -- (true) node[above] {True};
  \draw[->] (true) -- (body);
  \draw[->] (false) -- (cond); \end{tikzpicture}
\end{figure}
do ... while()
Topic 04: Loops (advanced)

- Equivalency
- More complex `for()` loops
- More advanced nesting
- `do ... while()` 4.4
- `break` and `continue` Special Topic 4.1
break;

• Can be used anywhere inside a loop
  – Works in `for()`, `while()`, and `do...while()`
  – Often (not always) inside an `if()` inside the loop

• Immediately terminates the loop
  – Jump out of the body without finishing it
  – Ignore the loop condition; just end
This Loop Isn't Actually Infinite...

```java
int i = 0;
while (true)
{
    if (i == stopAt)
        break;

    System.out.println(i);
    i++;
}
```
Why \texttt{break;}? 

- Sometimes, the top of the loop isn't a convenient place to check the loop condition:

\begin{verbatim}
while(true)
{
    int val = \ldots read input\ldots
    if (val == 0)
        break;

    \ldots do something with val\ldots
}
\end{verbatim}
Why \texttt{break;} ?

- Sometimes, you have two loop conditions, to check at different places:

```c
while (condition1)
{
    ...

    if (condition2)
        break;

    ...
}
```
continue;

- Just like `break;` **EXCEPT:**
  - Test the loop condition
  - Continue looping if the loop condition is valid
An Example of `continue`;

```java
System.out.println("2");
for (i=3; i<2*1000*1000*1000; i++)
{
    // all evens >= 2 are composite
    if (i % 2 == 0)
        continue;

    ... check to see if i is prime ...

    if (prime)
        System.out.println(i);
}
```
An Example of `continue`;

```java
System.out.println("2");

for (i=3; i<2*1000*1000*1000; i++)
{
    // all evens >= 2 are composite
    if (i % 2 == 0)
        continue;

    ... check to see if i is prime ...

    if (prime)
        System.out.println(i);
}
```

If a number is even, don't even both checking.

Just move on to the next value.
An Example of `continue;`

```java
System.out.println("2");
for (i=3; i<2*1000*1000*1000; i+=2) {
    ... check to see if i is prime ...
    if (prime)
        System.out.println(i);
}
```

Another way to do the same thing.
return; Inside a Loop

- Immediately return, don't finish the loop
  - Think: “break-then-return”

```c
int firstFactor(int val)
{
    for (int factor=2; factor < val; factor++)
        if (val % factor == 0)
            return factor;

    return val;
}
```
Topic 04: Loops (advanced)

- Equivalency
- More complex `for()` loops
- More advanced nesting
- `do ... while()`
- `break` and `continue`  

Summary

Special Topic 4.1