Some Java Details

Syntax

Control

etc.

```java
public class lect4 {

    //************************************************
    //  Lect 4 demo program
    //************************************************
    public static void main(String[] args) {
        int x,y,pre,post;
        x = 7;
        y = 7;
        pre = ++x;
        post = y++;
```

Increment order

```java
public class lect4 {

    //*************************************************
    //  Lect 4 demo program
    //*************************************************
    public static void main(String[] args) {
        int x,y,pre,post;
        x = 7;
        y = 7;
        pre = ++x;
        post = y++;

        System.out.println("pre contains:  " + pre +
                " and x ends up with " + x);
        System.out.println("post contains: " + post +
                " and y ends up with " + y);

    }

}
```

```java
public class lect4 {

    //*************************************************
    //  Lect 4 demo program
    //*************************************************
    public static void main(String[] args) {
        int x,y,pre,post;
        x = 7;
        y = 7;
        pre = ++x;
        post = y++;

        System.out.println("pre contains:  " + pre +
                " and x ends up with " + x);
        System.out.println("post contains: " + post +
                " and y ends up with " + y);

    }

}
```

```
pre contains:  8 and x ends up with 8
post contains: 7 and y ends up with 8
```

Increment order

```java
public class lect4 {

    //*************************************************
    //  Lect 4 demo program
    //*************************************************
    public static void main(String[] args) {
        int x,y,pre,post;
        x = 7;
        y = 7;
        pre = --x;
        post = y--;

        System.out.println("pre contains:  " + pre +
                " and x ends up with " + x);
        System.out.println("post contains: " + post +
                " and y ends up with " + y);

    }

}
```

```
pre contains:  6 and x ends up with 6
post contains: 7 and y ends up with 6
```

Decrement

```java
public class lect4 {

    //*********************************************
    //  Lect 4 demo program
    //*********************************************
    public static void main(String[] args) {
        int x,y;
        x = 12;
        y = 0;

        if ( (y != 0) & (x % y) == 0)
            System.out.println(y + " is a factor of " + x);
    }

}
```

```java
public class lect4 {

    //*********************************************
    //  Lect 4 demo program
    //*********************************************
    public static void main(String[] args) {
        int x,y;
        x = 12;
        y = 0;

        if ( (y != 0) & (x % y) == 0)
            System.out.println(y + " is a factor of " + x);
    }

}
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at lect4.main(lect4.java:12)
```

```java
public class lect4 {

    //*************************************************
    //  Lect 4 demo program
    //*************************************************
    public static void main(String[] args) {
        int x,y;
        x = 12;
        y = 0;

        if ( (y != 0) && (x % y) == 0)
            System.out.println(y + " is a factor of " + x);
    }

}
```

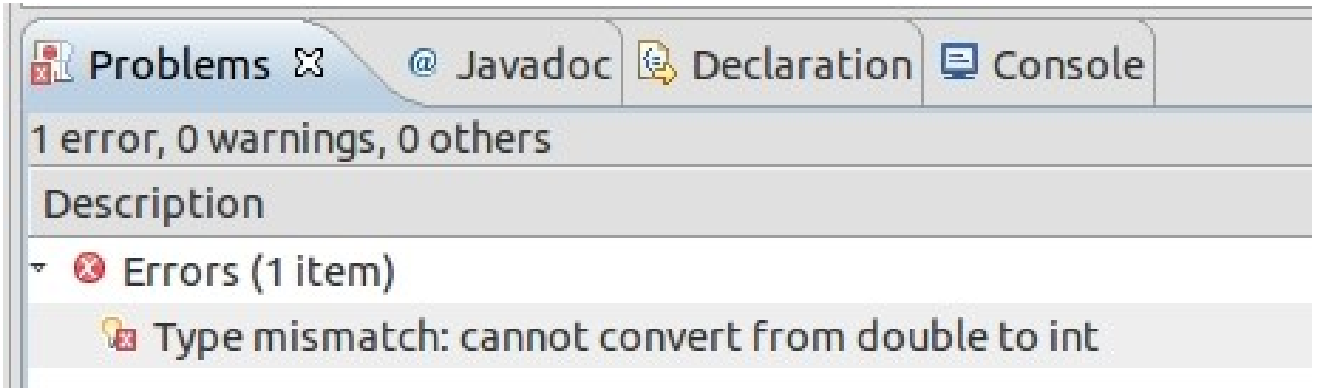
Conditional (short-circuit) logic AND operator

```java
public class lect4 {

    //************************************************
    //  Lect 4 demo program
    //************************************************
    public static void main(String[] args) {
        double x,y;
        int n;
        x = 12;
        y = 5;

        n = x/y;
    }

}
```



```java
public class lect4 {

    //***********************************************
    //  Lect 4 demo program
    //***********************************************
    public static void main(String[] args) {
        double x,y;
        int n;
        x = 12;
        y = 5;

        n = (int) (x/y);
    }

}
```

Type-cast syntax

Scope and lifetime of Variables

- Scope is a block denoted by { and }
- Variables can be declared within any block
- Scope determines object visibility to other parts of a program and also lifetime
- In general, variables declared inside a scope are not visible outside that scope (more details later)

- Scopes can be nested
  variables declared in outer nest are visible to inner
  variables declared in inner nest not visible to outer

- Variables are created when a scope is entered and destroyed when the scope is left.

Operation of Program Control Statements

```
if (expression) {
   ….
   }
else {
   ….
   }

if (expression) {
   ….
   }
else if (expression) {
   ….
   }
else {
   ….
   }
```

–-> work the same as C/C++

Program Control Statement Operation

```
switch(expression) {
    case constant1:
        statement sequence
        break;
    case constant2:
        statement sequence
        break;
  .
  .
    default:
        statement sequence
}
```

NOTE: Break needed

Allowed data types for the expression:

byte, short, int, char, enumeration (java 1.7+, string)

Program Control Statement Operation – continued

for loop ---> essentially the same as C/C++
loop executes only if the loop condition is true

there is a new for loop called enhanced for
it will be discussed later

Consider this for statement:

```
for (int i=0; I < 10; i++) {
    … loop body
}
```

What is the scope of variable i ?
(i.e what is the value of i after the loop completes?)

How many times is the loop body executed?

(The conditional expression is evaluated at the top of the loop)

Program Control Statement Operation – continued

for loop ---> essentially the same as C/C++
loop executes only if the loop condition is true

there is a new for loop called enhanced for
it will be discussed later

Consider this for statement:

```
for (int k=0; k < 10; ++k) {
    … loop body
}
```

What is the scope of variable k ?
(i.e what is the value of k after the loop completes?)

How many times is the loop body executed?

Program Control Statements - continued

while(expression) ---> same as C/C++
loop only executes if expression is true

do-while(expression) ---> same as C/C++
loop executes once for sure

Program Control Statements – continued

break; ---> similar to C/C++
break out of current loop level

break *label*; ---> break out of one or more blocks of code
could be a loop
can be any block of code
but
the block has to be labeled at its start

(is this a goto ?)

Read a character from the keyboard

```
class  demo {
     public static void main(String args[])
          throws java.io.IOException {

          char ch;

          ch = (char) System.in.read();       // get a character
          . . . . .
          do {                                          // clear buffer
               ignore=(char) System.in.read();
          } while(ignore != '\n');
          . . . . .
     }
}
```