Selection

Comparison Operators in Java

- = = Equals
- != not equal to
- > greater than
- < less than
- >= greater than or equal to
- <= less than or equal to

Syntax – the if statement

```
if ( <boolean expression> )
{
<operations to completed if boolean
   expression is true>
}
```

if..else if

- Remember: the first condition found true is the only body executed... automatic skip to after construct.

Syntax

```
if ( <boolean expression> ) {
<operations to completed if boolean expression is true>
}
else if ( <boolean expression> ){
<operations to completed if boolean expression is true>
}
```

else

- Remember: the first condition found true is the only body executed... automatic skip to after construct until the else statement where all remaining conditions are found true and is executed.

Syntax

```
if (boolean expression) {
<operations to completed if boolean expression is true>
}
else {
<operations to completed if all other selection statements
    are false>
}
```

Recall: the *if* statement

```
int number1 = 4;

if  (number1 == 0) {
        System.out.println ( “Batman” );
}
else if  (number1 == 1) {
        System.out.println ( “Superman” );
}
else if  (number 1 == 2) {
        System.out.println ( “Wonder Woman” );
}
else if (number1 == 3) {
        System.out.println ( “Green Lantern” );
}
else {
        System.out.println ( “Flash” );
}
```

OUTPUT
Flash

boolean operators

- There are two commonly used boolean operators in Java to **combine** conditions

- && , which means **and**
- || , which means **or**

Consider the following:

- To attain a letter grade of “B” in this class you must attain 70% - 80% exclusive.
- That is:

  If mark is >= 70 AND mark < 80

  You will attain a “B”
- In the previous example you will attain a “B” ONLY if BOTH conditions are met.
  - That is: mark is both >=70 and <80

&&: An example

```
if (mark>=70) && (mark<80) {
    System.out.println(“You have a B”);
}
```

Syntax Switch

- To avoid the *if..else if* multiple-selection statements, Java provides the *switch multiple-selection* statement to make your code a little more legible.

Syntax

```
switch ( <switch variable> )
{
      case <case variable>:
            <operations to completed if boolean expression is true>
            break; // necessary to exit the switch statement
      case <case variable>:
            <operations to completed if boolean expression is true>
            break;
      ……..
      default:
            <operations to completed if boolean expression is true>
            break;
}
```

How it works

- The switch variable is compared to **each** case variable. If they are equal (i.e. true), then the code below the applicable case statement is executed
- The **break** is optional and is necessary to exit the switch statement if so desired once a true evaluation is found
  - Without any break statements, all subsequent code blocks will be executed once a true evaluation is found

An Example

```
int number1 = 3;

switch (number1){
	case 0:
		System.out.println ("Batman");
		break;
	case 1:
		System.out.println ("Superman");
		break;
	case 2:
		System.out.println ("Wonder Woman");
		break;
	case 3:
		System.out.println ("Green Lantern");
		break;
	default:
		System.out.println ("Flash");
}
```

OUTPUT

Let's remove the break statements

```
int number1 = 2;

switch (number1){
        case 0:
                System.out.println ("Batman");
        case 1:
                System.out.println ("Superman");
        case 2:
                System.out.println ("Wonder Woman");
        case 3:
                System.out.println ("Green Lantern");
        default:
                System.out.println ("Flash");
}
```

OUTPUT
Wonder Woman
Green Lantern
Flash