Operators

Comparison Operators

Comparison operators

- return a boolean value (`true` or `false`)
- equals operator (a == b)
    - primitives: returns `true` if values are the same
    - objects: returns `true` if both values reference to the same object
- not-equals operator (a != b) works in the same way
    - primitives: returns `false` if values are the same
    - objects: returns `false` if both values reference to the same object
- you can only compare values of similar type (auto-casting applies)

```
// comparing objects

String name1 = new String("John Wayne");

String name2 = new String("John Wayne");

String name3 = name1;

System.out.println(name1 == name2);

System.out.println(name1 == name3);
```

name1 and name2 point to **different** objects => `false`

name1 and name3 point to **same** object => `true`

```
// if both references point to null, they are equal!
```

```java
// relational operators: <, <=, >, >=, instanceof

public class MyApp {

public static void isInteger (Number num) {

  // Integer is a subtype of Number

  if (num instanceof Integer)

    System.out.println(num + " is an integer.");

  else

    System.out.println(num + " is not an integer.");

  }

public static void main(String args[]) {

  isInteger(5);

  isInteger(3.14);

  }

}
```

```
5 is an integer.
3.14 is not an integer.
```

```
// you cannot use instanceof with unrelated types
// => it will not compile


// instanceof involving null
System.out.println(null instanceof Object);
  // => always returns false


System.out.println(null instanceof null);
  // DOES NOT COMPILE
```

Logical operators

- logical AND: `a & b`
  - true if at least one of the operands is true
- logical inclusive OR: `a | b`
  - true if at least one is true
- logical exclusive OR (XOR): `a ^ b`
  - true only if one value is true, another is false

Conditional operators

- conditional AND: a && b
  - true if at least one of the operands is true
- conditional OR: a || b
  - true if at least one is true

- what is the difference between logical and conditional operators?
  - conditional evaluation stops once the result can be determined
  - this property is known as *short-circuit*

```java
// example 1
int a = 5;
int b = 10;
int c = 15;
if ((a < b) || (++c > 10)) {
  System.out.println("We are in!");
}
System.out.println("c = " + c);
```

this is true, so the result of expression is true regardless of what is right of ||

this is never evaluated

```
We are in!
c = 15
```

this is true, but since this is a logical operator,

right side will be evaluated nevertheless

```
// example 2
int a = 5;
int b = 10;
int c = 15;
if ((a < b) | (++c > 10)) {
  System.out.println("We are in!");
}
System.out.println("c = " + c);
```

```
We are in!
c = 16
```

```java
// ternary operator also has short-circuit property
```

true never evaluated

```java
// example 1 (a=5, b=10, c=15)
int d = (a < b) ? 7 : ++c;
System.out.println("d = " + d + ", c = " + c);
```

```
d = 7, c = 15
```

false never evaluated

```java
// example 2
int d = (a > b) ? ++c : 12;
System.out.println("d = " + d + ", c = " + c);
```

```
d = 12, c = 15
```