Using if with numbers

Booleans, relational operators, numbers and decisions

The if statement (review)

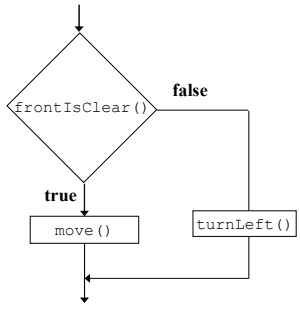
- The if statement can also be used with numbers
- Relational operators are used to compare numbers; they produce a condition or boolean result.
- Boolean expressions, boolean operators.

Remember the if statement

The if/else statement

```
if(frontIsClear())
   move();
else
   turnLeft();
```

If the front is clear, then the robot moves forward, otherwise it turns left.



The if statement

Syntax

```
if(condition)
   instruction;
else
   instruction;
```

Optional else part

Example

```
if(frontIsClear())
   move();
else
   turnLeft();
```

Purpose

To execute a statement when a condition is true or false.

The condition is a boolean value

Syntax

boolean

```
if(condition)
   instruction;
else
   instruction;
```

Optional else part

The methods from RobotWorld which could be used in an if statement all produced a boolean value, e.g.

```
frontIsClear()
facingNorth()
```

A boolean value is true or false

Boolean Expressions

- Boolean Expressions are a system for representing a mathematics of true and false
- Boolean expressions are called "Conditions"
- Boolean expressions are formed by comparing values with *relational operators* and combining boolean values with the *logical operators* || (or), && (and), and ! (not)

Example Boolean Expressions

```
lowerLimit > 5
applicantsAge <= 65
```

Relational Operators

Relational operators take two numbers and produce a boolean value.

Maths	*Java*	*English*
=	==	*equal to*
≠	!=	*not equal to*
<	<	*less than*
≤	<=	*less than or equal to*
>	>	*greater than*
≥	>=	*greater than or equal to*

Exercise 1

- What are the values of the following boolean expressions (assume age is an int):

	age is 20	age is 70
`age == 20`	T	
`age != 10`	T	
`age > 30`	F	
`age <= 65`	T	

Write down the values of the expressions assuming age = 70.

Exercise 2

- What will the following code fragment print

```
int x, y; // define x and y to be integers

x = 7;
y = 9;

System.out.println(x < 7);
System.out.println(x != y);
System.out.println(y - 2 == x);
System.out.println(x >= y);
```

Combining boolean expressions

Let's say we wanted to check the condition

```
MINIMUMWAGE <= wages <= MAXIMUMWAGE
```

This is not a legal Java statement, because the relational operators are binary operators.

Instead, we combine two boolean expressions using an AND operator:

```
(MINIMUMWAGE <= wages) &&
                (wages <= MAXIMUMWAGE)
```

AND operator

Exercise 3

- What values of `day` will make these expressions true

```
day < 8
day >= 100
day >= 10 && day <= 100
day <= 0 || day >= 7
```

The result of `&&` is true only if both operands are true.

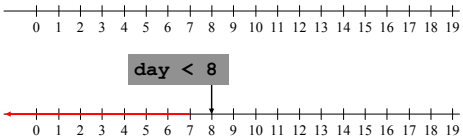
The result of `||` is true if either operand is true.

- Note for

`&&` read AND
`||` read OR
`!` read NOT

The number line

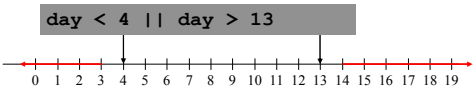
In these situations it can help to think of the number line



All the numbers less than 8 will make the condition true.

The number line

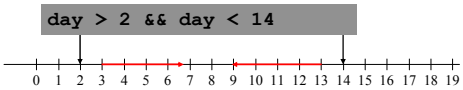
The number line can help when dealing with more complex boolean expressions.



All the numbers less than 4 or greater than 13 will make the condition true; this means 3, 2, 1 ... as well as 14, 15, 16 ...

The number line

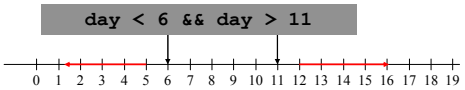
What about AND?



This condition is true for any of the numbers 3, 4, 5, ... 12, and 13.

if

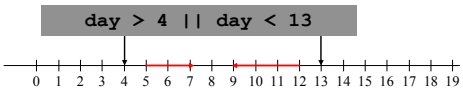
What values of day will make this condition true?



This condition is true for no values.

if

What values of day will make this condition true?



This condition is always true. No matter what number you think of, it will be either less than 13, or greater then 4

Exercise 4

Write an expression that checks to see if an age is greater than or equal to 18. It might be used in the following code fragment:

```
int age;
   :
   :
if( ... )
   System.out.println("You can vote!");
```

Exercise 5

Write an expression that checks to see if an age is that of a teenager (13 to 19 inclusive). It might be used in the following code fragment:

```
int age;
   :
   :
if( ... )
   System.out.println("Is a teenager");
```

Boolean variables

A boolean variable can be assigned to a boolean expression

```
boolean isTeenager;

isTeenager =  age > 12 && age < 20;

if(isTeenager)
   System.out.println("Is a teenager");
```

if age > 12 **AND** age < 20 then the result is true. This is assigned to the variable.

Remember precedence

Parentheses (brackets)	*Highest precedence*
Unary operators + − !	
Binary arithmetic operators * / %	
Binary arithmetic operators + −	
Relational operators < > <= >=	
Relational operators == !=	
Logical operator &&	
Logical operator ||	*Lowest precedence*

Precedence in a boolean expression

```
isTeenager = age > 12 && age < 20;
```

The relational operations are carried out first. The two boolean values are compared using the **&&** operator to produce the boolean result.

Example

```
public class IfTest
{
   public static void main(String [] args)
   {
      int age;
      System.out.println("Enter age: ");
      age = Console.readInt();
      if(age < 6)
      {
         System.out.println("That's young");
      }
   }
}
```

Example 2

```
public class IfTest2
{
   public static void main(String [] args)
   {
      int temperature;
      System.out.println("Enter temperature: ");
      temperature = Console.readInt();
      if(temperature < 0)
      {
         System.out.println("It's cold");
         if(temperature > 35)
            System.out.println("It's hot");
         else
            System.out.println("It's OK");
      }
   }
}
```

Summary

- The if statement needs a boolean value (or condition)
- Relational operators are used to compare numbers; they produce a condition or boolean result.
- Evaluating boolean expressions, boolean operators, `&& || !`.

Questions?