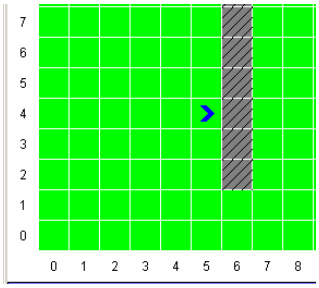
The if statement

- Making decisions
- The boolean type has two values
  - true
  - false

Consider the Robot below

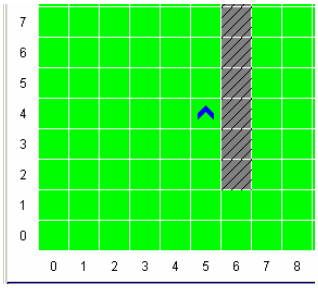


Can the Robot move forward without crashing?

Is the cell in front of the Robot clear or not?

The answer to this question has to be either **true** or **false**

What about this Robot?



Now is the cell in front of the Robot clear or not?

Conditions

- Let's call the condition of the cell being empty *frontIsClear*. It is either **true** or **false**. Such a value is called a **boolean** value.
- Robots have various boolean methods that they can use to find out about the world.
- One such method is frontIsClear()

Robots' boolean methods

```
boolean frontIsClear()
boolean facingNorth()
boolean facingSouth()
boolean facingEast()
boolean facingWest()
boolean hasBeeper()
boolean beeperPresent()
```

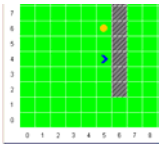
Move

Direction

Beepers

Question

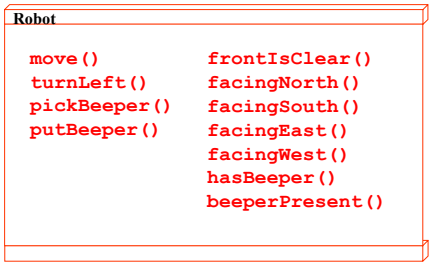
```
boolean frontIsClear()
boolean facingNorth()
boolean facingSouth()
boolean facingEast()
boolean facingWest()
boolean hasBeeper()
boolean beeperPresent()
```



What values will these methods give for the above Robot?

Assume the Robot executes a turnLeft() instruction. Which methods will now give a different value?

The Robot class



The if statement

The **if** statement executes an instruction only if a condition is true

For example:

```
if(frontIsClear())
   move();
```

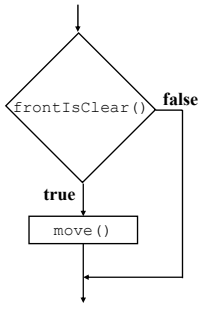
The condition **`frontIsClear()`** is checked. If it is true, the following instruction is executed

This allows the robot to check that the way is clear before moving.

The if statement

The if statement can be represented by a flowchart

```
if(frontIsClear())
   move();
```

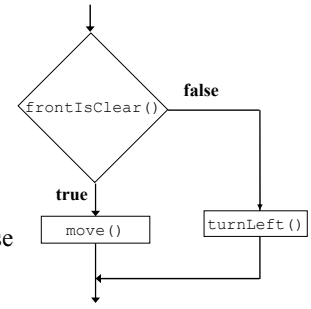


else

The if statement can have an else part

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

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



Syntax: if statement

Syntax

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

Optional else part

Example

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

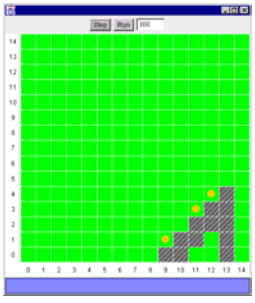
Purpose

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

Example

Remember the stairs example; bill would climb the stairs and pick up the beepers. But if a beeper was missing an error would occur.

No problem: use an if statement to see if a cell contains a beeper



Example

The following instructions check whether a beeper is present before attempting to pick it up.

```
if(beeperPresent())
   pickBeeper();
move();
```

If true, bill picks up the beeper and then executes the move() method

If false, bill skips the pickBeeper() method and just executes the move() method.

Indentation

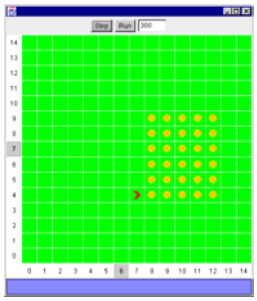
- Every instruction that is controlled by an if statement should be indented by three spaces.
- For example

```
if(beeperPresent())
   pickBeeper();
move();
```

This is correctly indented

Field Of Beepers

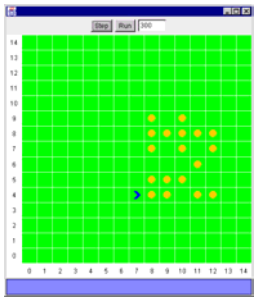
Recall the problem of getting a robot to sweep up a field of beepers. Now imagine if there wasn't a beeper on every square.



The Field with Gaps

You would have to change the program to check if the cell contained a beeper before attempting to pick it up.

Every time you go to pick up a Beeper, you must check if there is one there.



If the program used methods

Just change one method from

```
void moveAndPick()
{
   move();
   pickBeeper();
}
```

to

```
void moveAndPick()
{
   move();
   if(beeperPresent())
      pickBeeper();
}
```

This new method ensures you won't try and pick up a Beeper if none is present

Just the Opposite

- Sometimes we want the opposite condition.
- For example, if frontIsClear() is false, then we would like to turn left.
- For this situation, Java provides the NOT operator, **!**, an exclamation mark.
- If frontIsClear() is **true**, then !frontIsClear() is the opposite, i.e. **false**.

Not Example

```
if(!frontIsClear())
   turnLeft();
```

Whenever, you see an exclamation mark in Java, read NOT

This should be read as follows:
If NOT frontIsClear() then
  turn left

A Block of Instructions

```
void turnAroundIfBlocked()
{
   if(!frontIsClear())
   {
      turnLeft();
      turnLeft();
   }
}
```

These braces are used to create a block of instructions

The **if** statement normally only affects one instruction. If we want to operate on more instructions, we create a block by surrounding the instruction with braces

Forgetting the braces

```
void turnAroundIfBlocked()
{
   if(!frontIsClear())
      turnLeft();
      turnLeft();
}
```

The indentation misleads us. What does this method do?

Face North

```
void faceNorth()
{
   if(!facingNorth())
      turnLeft();
   if(!facingNorth())
      turnLeft();
   if(!facingNorth())
      turnLeft();
}
```

This method will cause the robot to turn to the north regardless of the original direction the robot was facing

Nested if

An "if" instruction that is itself controlled by another if statement, for example

```
if(frontIsClear())
{
   if(facingNorth())
   {
      move();
   }
}
```

Nested if

Nested if/else

When a nested if statement has an else part, the else always matches the nearest incomplete if statement

```
if(frontIsClear())
{
   if(facingNorth())
   {
      move();
   }
   else
   {
      turnLeft();
   }
}
```

Transforming if statements

The following two code fragments are equivalent

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

```
if(!facingNorth())
   move();
else
   turnLeft();
```

In general, we can turn one if/else into an equivalent one by:
1. Replacing the condition with its opposite, and
2. Interchanging the two statements

Summary

- A boolean value is either true or false.
- Using boolean methods and the if statement a Robot can make decisions.
- The if statement checks a condition. The following instruction is only executed if the condition is true.
- The not operator, !, reverses the condition.
- Use braces to create an instruction block.
- Use indentation to make your programs clearer.

Questions