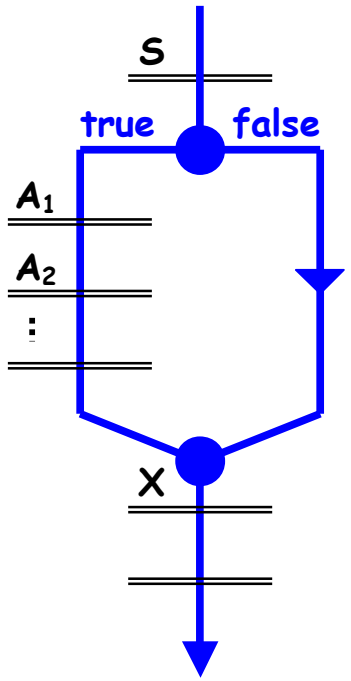
Introduction to Java
Part 2

LASSONDE SCHOOL OF ENGINEERING | YORK UNIVERSITÉ UNIVERSITY | EECS ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Branching: if-Statement

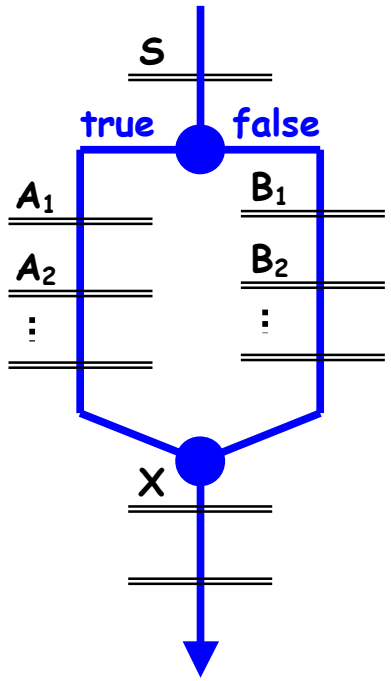
- Involves the evaluation of a Boolean expression
- If the expression evaluates to true, code execution takes a separate path
- In Java, the separate path is enclosed in a code block (indicated by braces)
- If the expression evaluates to false, code execution continues with the statement after the code block

if Statement




```
Statement-S
if (condition)
{
    Statement-A1
    Statement-A2
      ⋮
}
Statement-X
  ⋮
```

if-else Statement



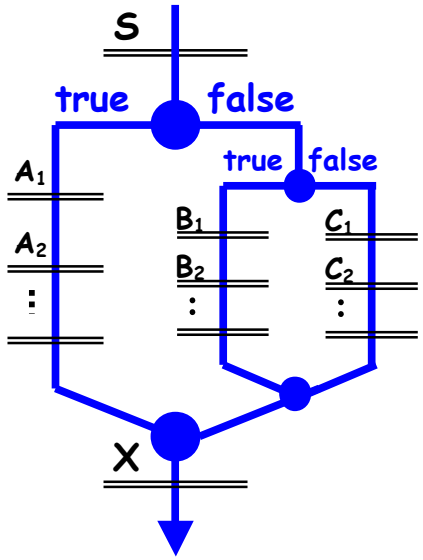

```
Statement-S
if (condition)
{
    Statement-A1
    Statement-A2
    ...
} else
{
    Statement-B1
    Statement-B2
    ...
}
Statement-X
```

if or if-else?

- Beneficial to use “if-else” statements
  - Clearly represents “decision making” choices
  - Aids in debugging logic errors

- Better to use “if” statements if “else” block is empty

Multiple Conditions (if, else-if, else)




```
Statement-S
if (condition-1)
{  Statement-A1
   Statement-A2
   ...
} else if (condition-2)
{  Statement-B1
   Statement-B2
   ...
} else
{  Statement-C1
   Statement-C2
   ...
}
Statement-X
```

Exercise: Even or Odd

- Write a program that determines if the value of an int variable is even or odd

- Example output:
  - The number 6 is even.
  - The number 5 is odd.

- Code demonstrated in lecture

Pitfall: Including a Semicolon

- Example

```
int entry = 3;
int absValue = entry;
if (entry < 0); ← wrong!
{
  absValue = -entry;
}
System.out.println(absValue);
```

- Consequently, the entry will always be negated

Pitfall: Omission of Braces

- Example

```
if (count > maximum)
        count--;
        System.out.println("Maximum exceeded.");
```

- Count will be decremented if the condition is true

- Print statement will be executed regardless

Pitfall: Variable out of Scope

- Variables declared in a code block are accessible only within that block

- Example

```
if (entry < 0)
{
   int absValue = -entry;
}
System.out.println(absValue);
```

- Variable absValue is not accessible outside the block → results in a compile time error

Application Programming Interface (API)

- Documents the pre-made classes that are available to Java programmers
- Describes each class, its features, and sometimes gives examples on its use
- Wealth of information, but sometimes overwhelming and difficult to read
  - Important to find what you need

API Layout

Packages	Details
	The Class section
	The Field section
Classes	The Constructor section
	The Method section

API - Fields

Field Summary

`static double`	**PI** The `double` value that is closer than any other to *pi,* the ratio of the circumference of a circle to its diameter.

Field Detail

PI

```
public static final double PI
```

The double value that is closer than any other to pi, the ratio of the circumference of a circle to its diameter.

See Also: Constant Field Values

Overloading Methods

- Method signature
  - Method name + parameter types
  - Must be unique within a class regardless of return type
- Overloaded methods
  - Same name, but take different parameters
- Example
  - Math.abs(double a)
  - Math.abs(long a)

Utility Classes

- Perform computation, not data storage
- Represent computations, not objects
- E.g., Math class
- All methods and attributes are **static**
  - Can be called without first declaring an object
  - E.g., Math.PI, Math.E, Math.round(), Math.log()
- Non-utility classes may also have some static methods and/or attributes

Exercise: Power Calculator

- Write a program that calculates the power of a base raised to an exponent
- The values of the base and exponents are stored in variables
- The output is rounded to one decimal place
- Code demonstrated in lecture

Creating Classes

- Classes are the blueprint for objects – objects that represent real-world concepts
- Create a class that represents a bank account
- What attributes should it have?
  - What information is associated with your account?
- What methods should it have?
  - What can you do with your account?

BankAccount Class

- Code demonstrated in lecture