`if` Control Structure

Introduction to Programming

1

Control Structures

Control Structures determine the order in which statements or instructions are carried out in a Java program.

The three main control structures in any programming language are

- Sequence
- Selection
- Repetition/Iteration

2

Control Statements

Sequence - instructions are executed in the same order in which they appear within the program

- Unrealistically simple – no logical control structures

Selection - one group of statements is selected from several available groups

Repetition - instructions are executed *repeatedly*, until some logical condition has been satisfied.

3

Selection

Programs often require alternative actions depending on different inputs to the program by the user.

For example, the price of a cinema ticket may be determined by age.

PROGRAM OUTPUT 1

```
Please enter age:23
Price per ticket: €10.00
```

PROGRAM OUTPUT 2

```
Please enter age:16
Price per ticket:€5.00
```

`if` selection structure

- The `if` selection structure is used to choose between alternative courses of action.
- This involves the keyword `if` followed by an *expression* in brackets.
- If the *expression* is found to be `true`, then the statements following the `if` are executed.
- If the *expression* is `false`, the statements following the `if` are not executed.

5

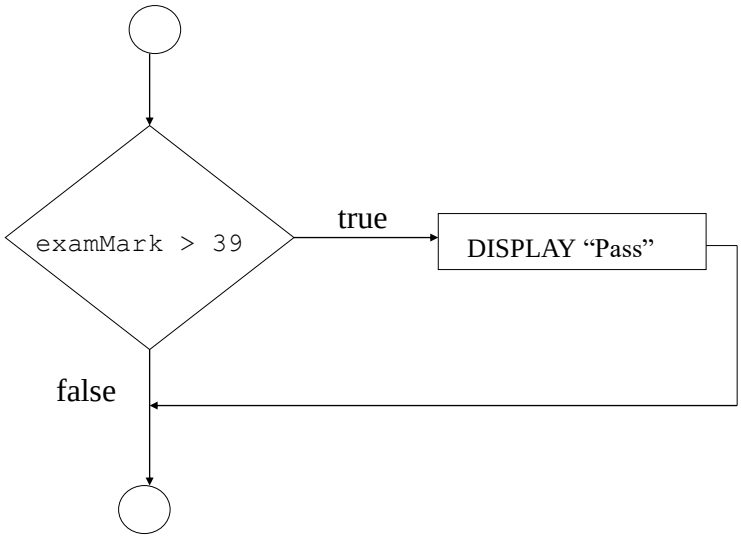
`if` example

- Determine if a student has passed an exam

```
if(examMark > 39)
{
   System.out.print("Pass");
}
```

6

Flow chart for *if* statement

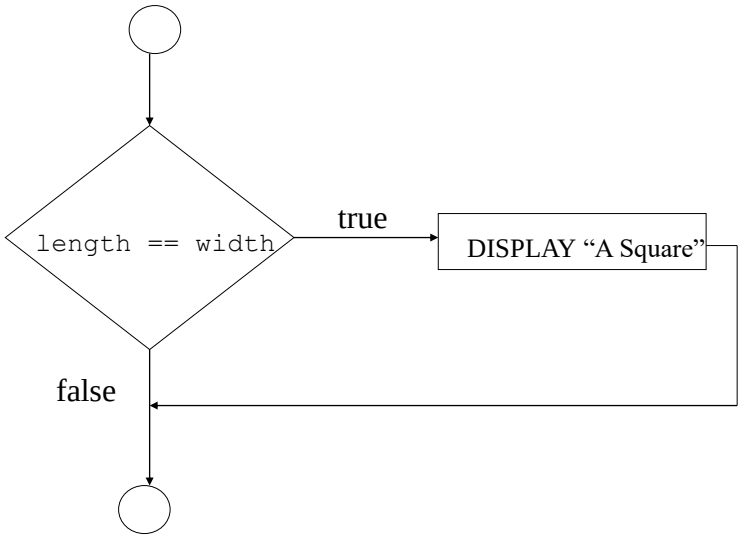


`if` example

- Determine if a rectangle is a square

```
if(length == width)
{
  System.out.print("it is square");
}
```

Flow chart for ***if*** statement



Logical expressions

- Selection depends on logical expressions that can be evaluated to `true` or `false`
- To do this we use *relational operators* and *equality operators*
- Also Known as conditional / logical / control / boolean expressions

Relational operators

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

Equality operators

= =	equal to
!=	not equal to

11

Comparing Values: Relational Operators

Relational operators compare values

Java	Math Notation	Description
>	>	Greater than
>=	≥	Greater than or equal to
<	<	Less than
<=	≤	Less than or equal to
= =	=	Equal to
!=	≠	Not equal to

- The == denotes equality testing

```
a = 5; // Assign 5 to a
if (a == 5) . . . // Test whether a equals 5
```

12

Common error

Confusing equality (==) Operator and Assignment Operator (=)

If these operators are confused it will not always result in a compiler error - the program may compile successfully but it will not run as expected.

Relational operators

Given the following variable declarations evaluate the expressions below

```
int  i = 1, j = 2, k = 3;
```

Expression	***Interpretation***
i < j	
(j+ k) > (i + 5)	
k != 3	

3 Space Indentation

3 space indentation is important in the `if` statement. Everything after the if statement should be indented 3 spaces (use tab) as shown.

```
if(grade > 39)
{
~~~System.out.print("PASS");
}
```

`if else` statement

- The `if` else statement allows us to execute different statements when the `if` expression evaluates to `false`.
- The program will run one or other set of statements, depending on circumstances.

Example : The `if else` Statement

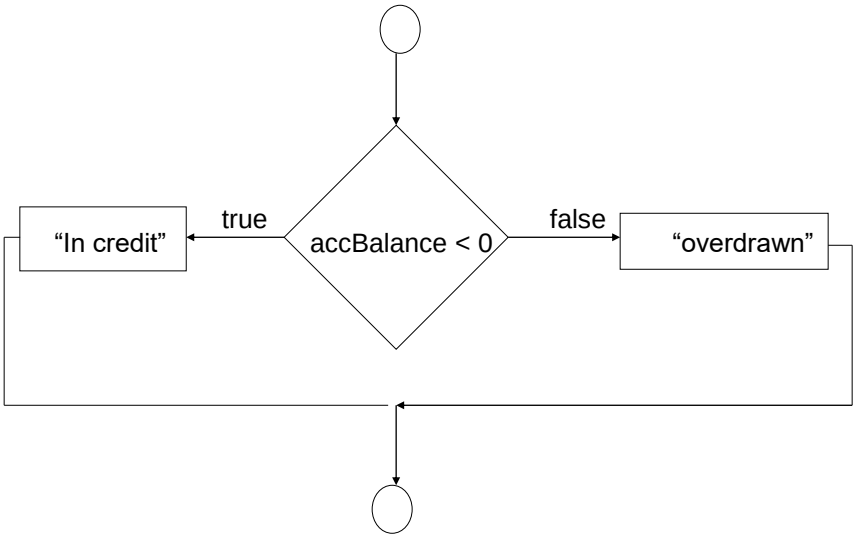
```
if (balance >= 0)
{
 System.out.println("In credit");
}
else
{
    System.out.println("Overdrawn");
}
```

If `balance >= 0` is *true*, then the *target* of the ***if*** will execute, and the ***else*** portion will be skipped

If `balance >= 0` is *false*, then the *target* of the ***if*** is bypassed and the *target* of the ***else*** will execute

Under no circumstances will both statements execute

Flow chart for `if else` statement



`if - else` Example

```
//determine price depending on age
if(age < 12 )
{
  ticketPrice = 3.50;
}
else
{
  ticketPrice = 8.00;
}
//rest of program
System.out.print("Amount Due: "+ticketPrice);
```