Selection
Chapter 3 – Part1

Course: CPSC 1150
Instructor: Dr. Bita Shadgar

Lecture 7

Learning Outcomes

- Implement conditions to make decisions using relational operators
- Implement complex conditions using Boolean operators
- Evaluate complex conditions using precedence operators
- Generate random numbers in Java

Programming Structures

Definition

Sequence structure is a set of statements that execute in the order they appear

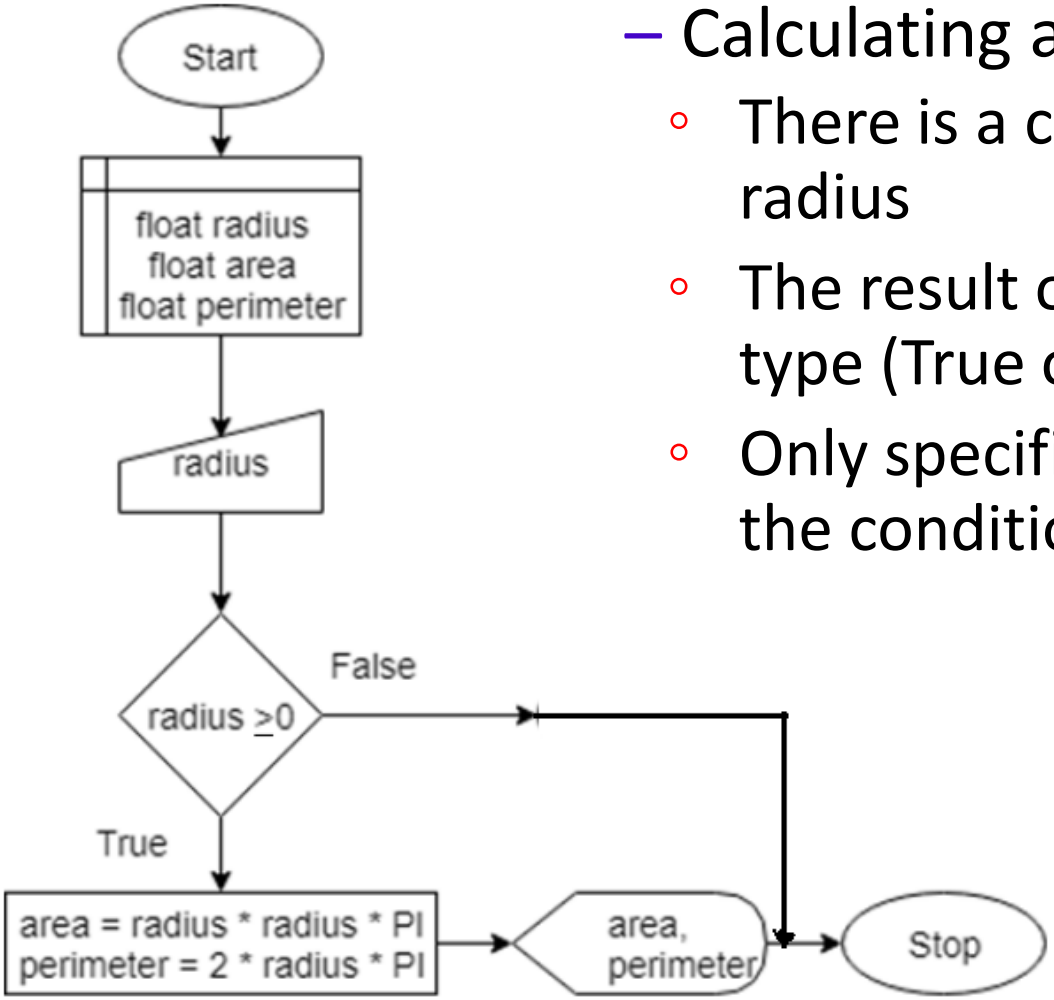
Definition

Control structure is a logical design that controls the order of executing a set of statements

Definition

Decision structure: is a kind of control structure that performs specific action(s) only if a condition exists. It is also known as selection structure.

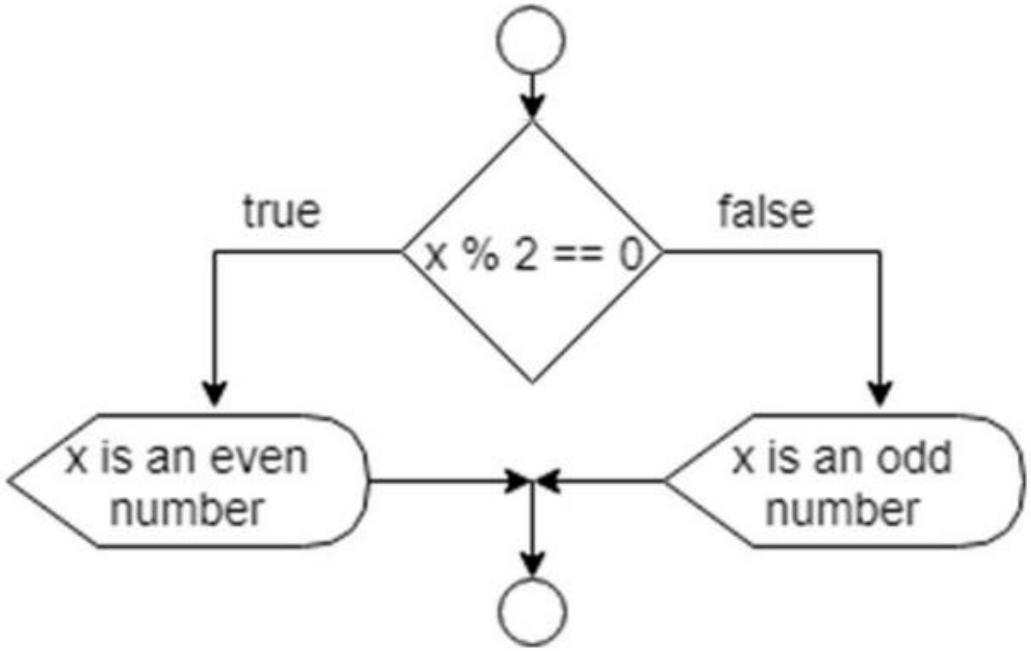
Example – Circles



- Calculating area and perimeter
  - There is a condition on the value of radius
  - The result of condition is a Boolean type (True or False)
  - Only specific instruction(s) executed if the condition exists

Decision Structure

- In flowchart, diamond represents true/false condition that must be tested
- Actions can be conditionally executed



Conditions and relational operators

Definition

Condition is a Boolean expression using relational operators

– Relational operator (comparison operator)

Operator	Description	True Example	False Example
<	Less than	3 < 8	8 < 3
>	Greater than	4 > 2	2 > 4
==	Equal to	7 == 7	3 == 9
<=	Less than or equal to	5 <= 5	8 <= 6
>=	Greater than or equal to	7 >= 3	1 >= 2
!=	Not equal to	5 != 6	3 != 3

Complex Conditions and Logical Operators

- Logical operators are used to create complex Boolean expressions
  - binary operators :
    - AND operator (&&)
    - OR operator (II)
    - XOR operator (^)
  - unary operator:
    - NOT operator (!)

The AND operator (&&)

- Takes two Boolean expressions as operands
  - Creates compound Boolean expression that is true only when both sub expressions are true
  - Can be used to simplify nested decision structures
- Truth table for AND operator (&&)
  - Example:

x >= 10 && x <= 20

A	B	A && B
false	false	false
false	true	false
true	false	false
true	true	true

The OR Operator (||)

- Takes two Boolean expressions as operands
  - Creates compound Boolean expression that is true when either of the sub expressions is true
  - Can be used to simplify nested decision structures
- Truth table for OR operator (||)
  - Example:
  - x < 10 || x > 20

A	B	A || B
false	false	false
false	true	true
true	false	true
true	true	true

The XOR Operator (^)

- Takes two Boolean expressions as operands
  - Creates compound Boolean expression that is true when both sub-expressions are different
- Truth table for XOR operator (^)
  - Example: The topic of Cryptography makes rich use of XOR function
  - To swap two variables like x and y without using temp.

```
x = x^y
y = x^y
x = x^y
```

A	B	A ^ B
false	false	false
false	true	true
true	false	true
true	true	false

The NOT Operator (!)

- Takes one Boolean expressions as operand and reverses its logical value (unary operator)
  - Sometimes it may be necessary to place parentheses around an expression to clarify to what you are applying the not operator
- Truth table for the NOT operator (!)
  - Example: !(x > 10)

A	!A
true	false
false	true

Short-Circuit Evaluation

Definition

Short circuit evaluation decides the value of a compound Boolean expression after evaluating only one sub expression, performed by the || and && operators

- For || operator
  - If the first operand is true, the result is true. Otherwise, evaluate right operand
- For && operator
  - If the first operand is false, the result is false. Otherwise, evaluate right operand

Operator precedence

1. var++, var–
2. +, - (Unary plus and minus), ++var,--var
3. (type) Casting
4. ! (Not)
5. *, /, % (Multiplication, division, and remainder)
6. +, - (Binary addition and subtraction)
7. <=, >, >= (Comparison)
8. ==, != (Equality)
9. ^ (Exclusive OR)
10. && (Conditional AND) Short-circuit AND
11. || (Conditional OR) Short-circuit OR
12. =, +=, -=, *=, /=, %= (Assignment operator)

Operator Precedence (cont'd.)

- Two important conventions
  - The order in which operators are used makes a difference
  - Always use parentheses to change precedence or make your intentions clearer

```
// Assigns extra premiums incorrectly
if(trafficTickets > 2 || age < 25 && gender == 'M')
   extraPremium = 200;
```

The expression that uses the && operator is evaluated first.

```
// Assigns extra premiums correctly
if((trafficTickets > 2 || age < 25) && gender == 'M')
extraPremium = 200;
```

The expression within the inner parentheses is evaluated first.

Definition

Operator associativity determines the order of evaluation, when two operators with the same precedence are evaluated.

- All binary operators except assignment operators are left-associative.
- Assignment operator is right-associative.

Example

- a – b + c – d is equivalent to ((a – b) + c) – d
- a = b *= c = 5 is equivalent to a = (b *= (c = 5))

Generating random numbers

- To generate a double random number, z, such that $0.0 \leq z < 1.0$, use the Math.random() method
- Often useful in programs
- Can algebraically manipulate the output of method to obtain a random output in another range

Random numbers in other range

```
r1 = (int) (Math.random() * 10);
//random integer between 0 and 9
r2 = 50 + Math.random() * 100;
//random double between 50 and 150(exclusive)
```

More Practice

- Write 3 different Boolean conditions using relational operators (choose different relational operators)
- Write 3 different Boolean conditions using logical operators (choose different logical operators)
- Write a Boolean condition that is opposite of “numbers bigger than 100”
- Generate a double random number between -93 and 77 (exclusive).