Java Programming

— Basics of Java Programming: Conditional branch —

Waseda University

Problem

Example

When height and weight are given (e.g. 172.0cm, 77.5kg), output the message according to the difference between Std. weight and given weight. Messages are the following:

Weight−Std. weight	Message
less than -10	`Underweight`
greater than or equal to -10, less than or equal to10	`Normal`
greater than 10	`Overweight`

Equation for calculating Std. weight:

$$\text{Std. weight(kg)} = \text{Height(m)}^2 \times 22$$

- For outputting suitable message, we need conditional branch
- There is `if statement` for selection control.

Problem

Example

When height and weight are given (e.g. 172.0cm, 77.5kg), output the message according to the difference between Std. weight and given weight. Messages are the following:

Weight−Std. weight	Message
less than -10	`Underweight`
greater than or equal to -10, less than or equal to10	`Normal`
greater than 10	`Overweight`

Equation for calculating Std. weight:

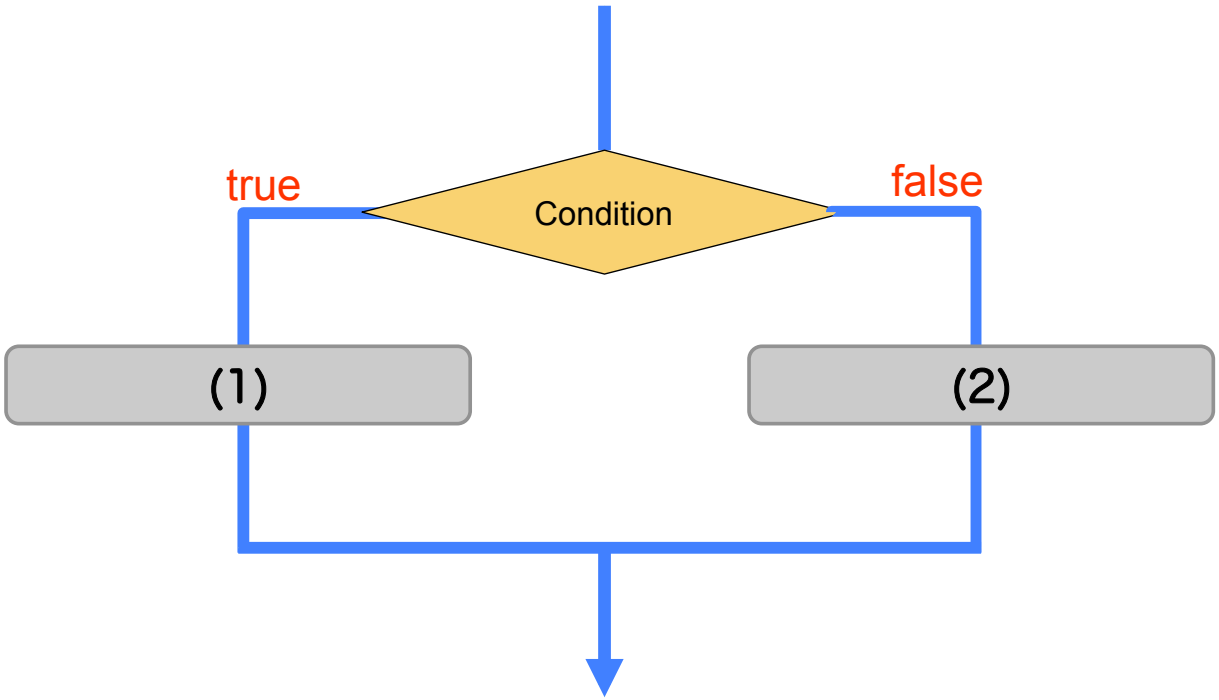
$$\text{Std. weight(kg)} = \text{Height(m)}^2 \times 22$$

- For outputting suitable message, we need conditional branch
- There is `if statement` for selection control.

Conditional branch

if-else statement

```
if (condition) {
 (1): statement when condition is true
} else {
 (2): statement when condition is false
}
```

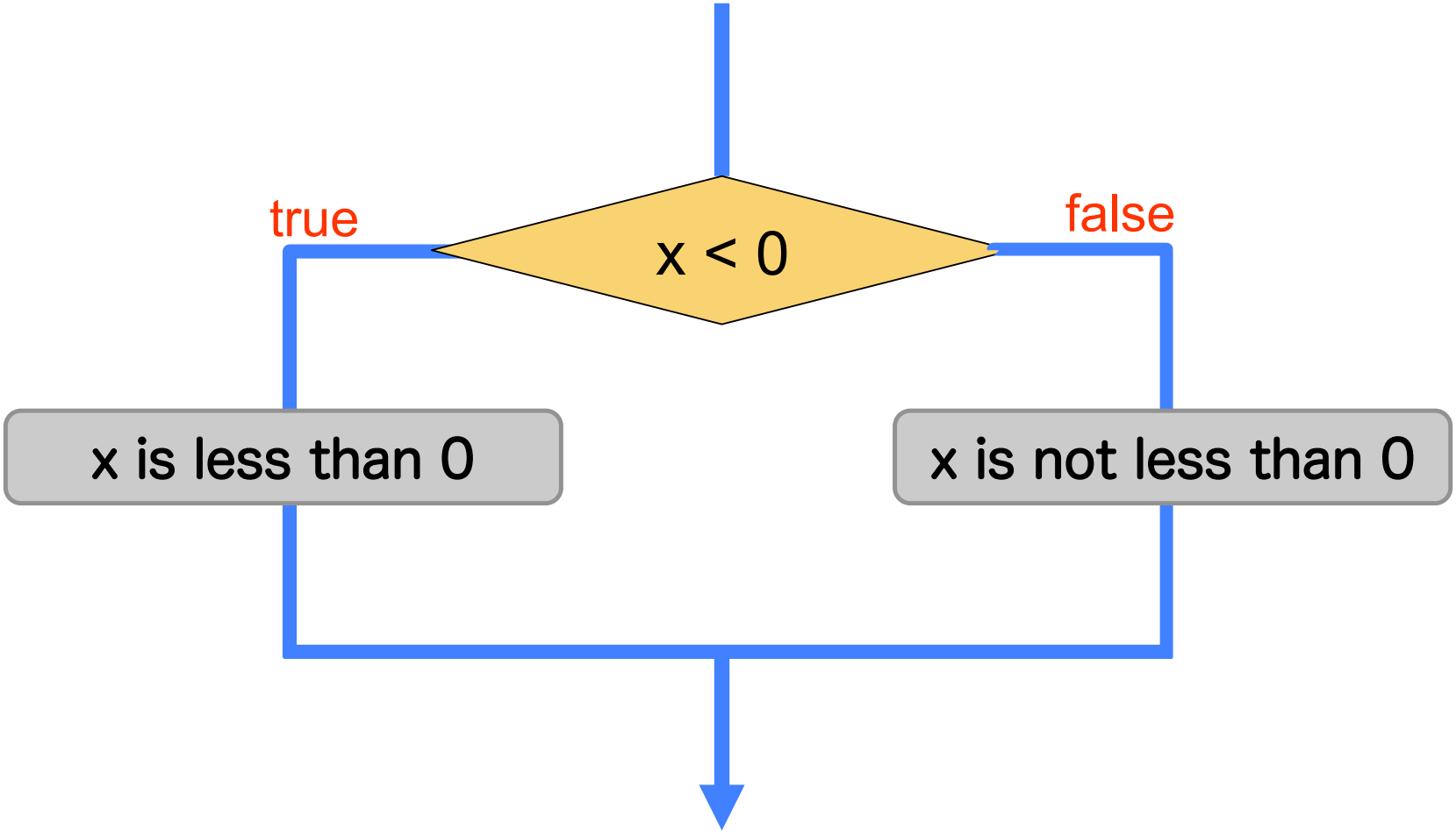


`if-else` **statement**

SampleIf.java

```
public class SampleIf {
  public static void main(String[] args) {
    int x = -5;
    if (x < 0) {
      System.out.println("x is less than 0.");
    } else {
      System.out.println("x is not less than 0.");
    }
  }
}
```

`if-else` **statement**

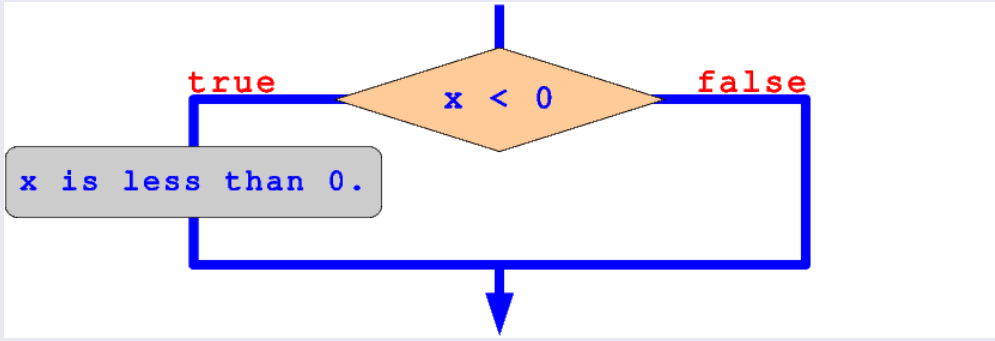


`if-else` statement

```
int x = *;
if (x < 0) System.out.println("x is less than 0.");
else       System.out.println("x is not less than 0.");
```

- When the statement in `{ }` is one, we can omit `{ }`.

```
int x = *;
if (x < 0) {
  System.out.println("x is less than 0.");
}
```



- We can omit after `else`.

Equality and relational operator

- Two numbers can be compared using the relational operators.
- In the relational expression,
  it is established → "`true`"
  it is not established → "`false`"
- Example of relational expression

```
3 < 5    → true
4 == 8   → false
```

Equality and relational operator

Operator	Meaning
==	Equality
!=	Inequality
<	less than
>	grater than
<=	less than or equal to
>=	greater than or equal to

Example

```
int a = 5;
System.out.println(a > 0);  → true
System.out.println(a <= 3); → false
System.out.println(a != 3); → true
```

Equality and relational operator

- Two numbers can be compared using the relational operators.
- In the relational expression,
  it is established → "`true`"
  it is not established → "`false`"
- Example of relational expression

```
3 < 5    → true
4 == 8   → false
```

Equality and relational operator

Operator	Meaning
==	Equality
!=	Inequality
<	less than
>	grater than
<=	less than or equal to
>=	greater than or equal to

Example

```
int a = 5;
System.out.println(a > 0);  → true
System.out.println(a <= 3); → false
System.out.println(a != 3); → true
```

Equality and relational operator

In case of example

```
if (diff < -10) {
  System.out.println("Underweight.");
}
```

- Variable `diff` : The difference between weight and Std. weight
- When `diff` is less than -10, it outputs `Underweight`.

Weight−Std. weight	Message
less than -10	`Underweight`
less than or equal to -10, greater than or equal to 10	`Normal`
greater than 10	`Overweight`

Equality and relational operator

"greater than or equal to -10 and less than or equal to10" ?

```
if (-10 <= diff <= 10) {
  System.out.println("Normal.");
}
```

←— This is wrong description

- By using logical operators, we can write it.

Logical operator

- The logical operators can be used to create a compound relational expression.
- Logical conjunction (`A && B`)

  `A and B`
  `A` and `B` is `true` → `true`, otherwise `false`.
- Logical disjunction (`A || B`)

  `A or B`
  Either `A` or `B` is true → `true`, otherwise `false`.
- Negation (`!A`)

  `A` is `true` → `false`, `A` is `false` → `true`

Logical operators

Operators	Meanig
&&	Conjunction
||	Disjunction
!	Negation

Logical operator

In case of example:

$-10 \leq \text{diff} \leq 10$

$\rightarrow$ "greater than or equal to -10" **and** "less than or equal to 10"

```
if (diff >= -10 && diff <= 10) {
  System.out.println("Normal.");
}
```

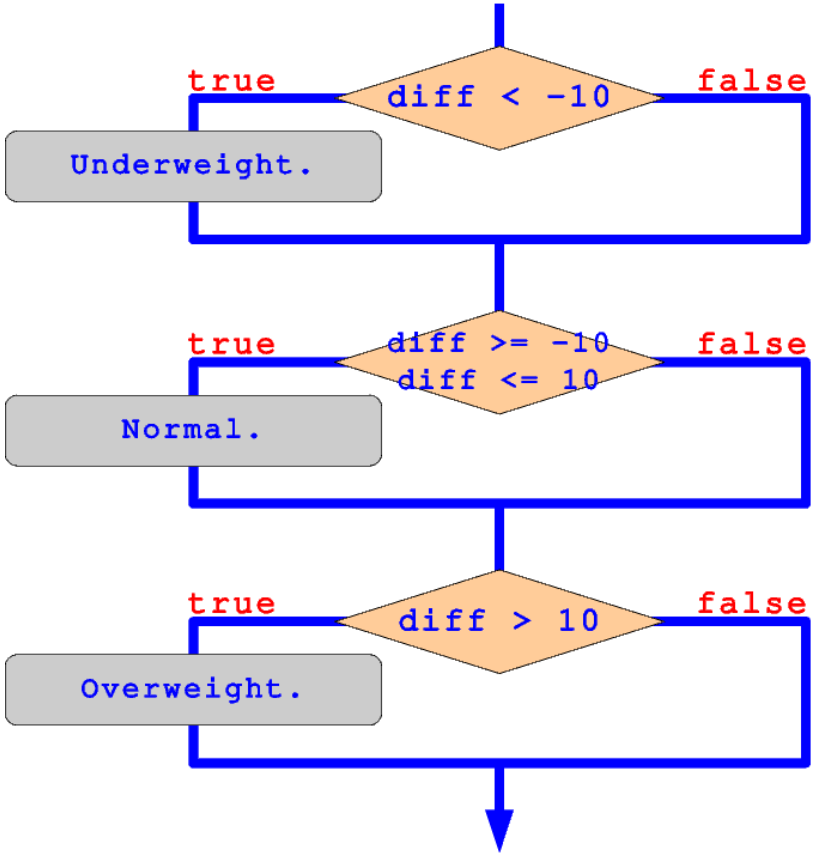
OR

```
if ((diff >= -10) && (diff <= 10)) {
  System.out.println("Normal.");
}
```

Example

By using relational and logical operators,

```
if (diff < -10) {
  System.out.println("Underweight.");
}
if ((-10 <= diff) && (diff <= 10)) {
  System.out.println("Normal.");
}
if (diff > 10) {
  System.out.println("Overweight.");
}
```

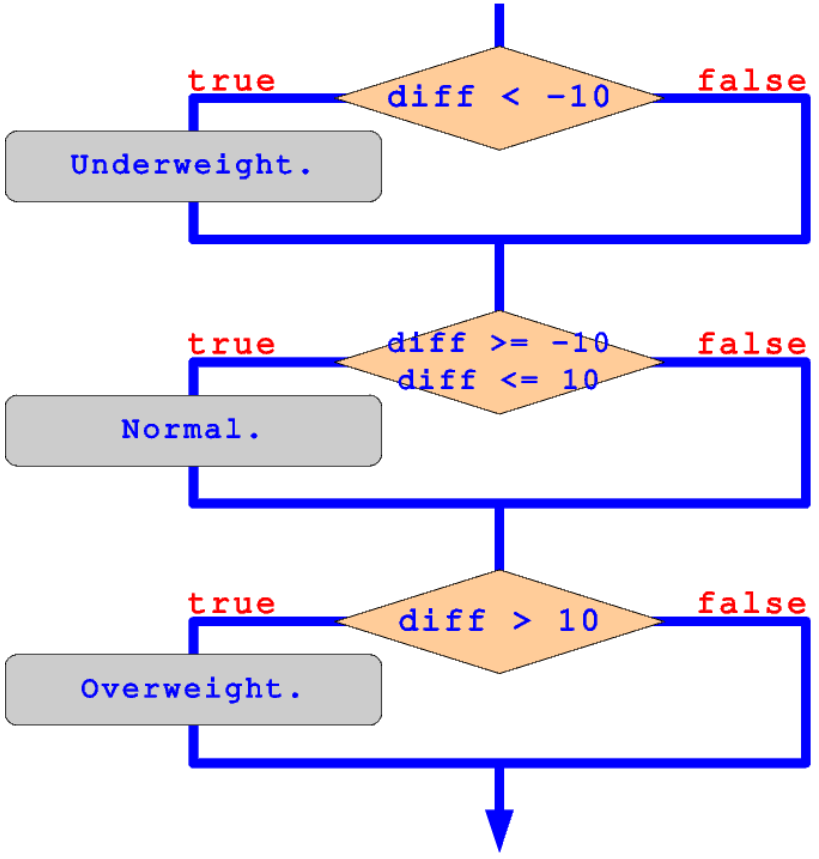


- It is simple, but it is wastefulness.
- 3 if statement are always processed.
- When `diff` is -12, first `if` statement is only congruent, but all `if` statement is performed.

Example

By using relational and logical operators,

```
if (diff < -10) {
  System.out.println("Underweight.");
}
if ((-10 <= diff) && (diff <= 10)) {
  System.out.println("Normal.");
}
if (diff > 10) {
  System.out.println("Overweight.");
}
```



- It is simple, but it is wastefulness.
- 3 if statement are always processed.
- When `diff` is -12, first `if` statement is only congruent, but all `if` statement is performed.

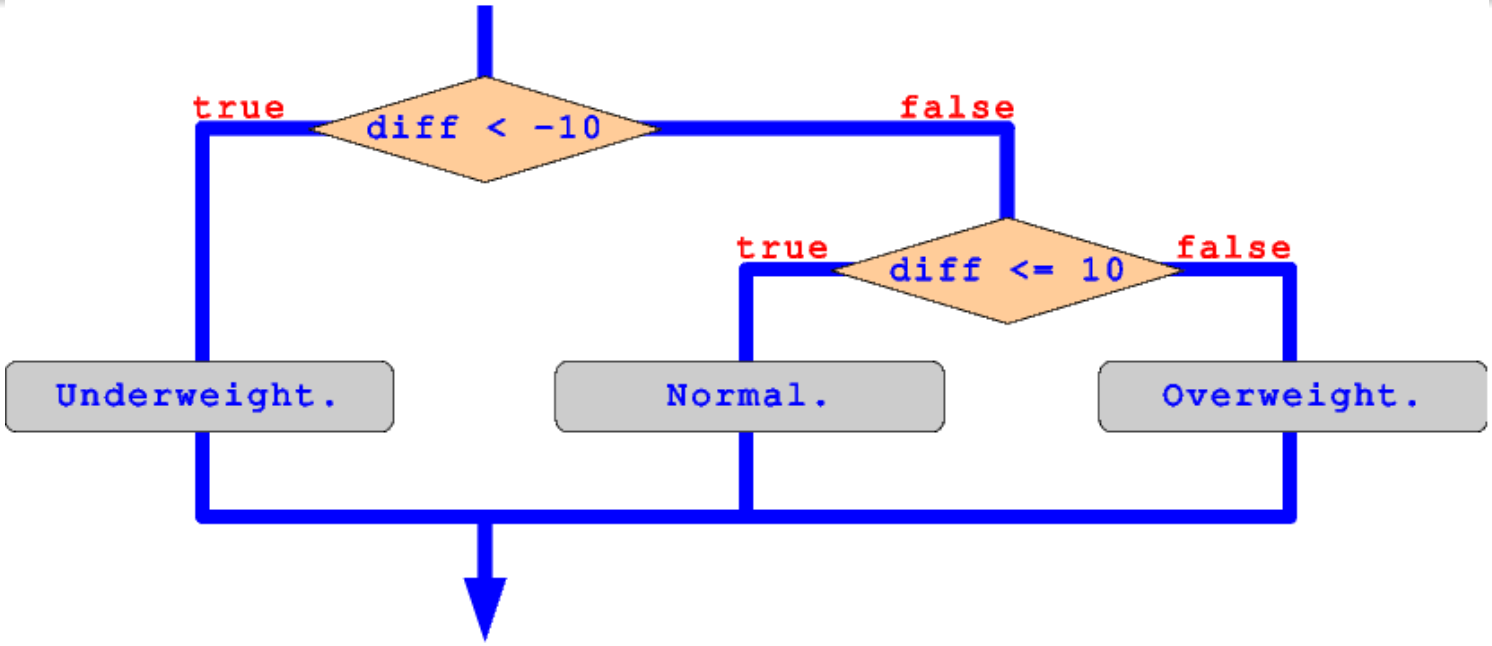
`else if` **statement**

```
if (condition1) {
    statement when condition1 is true
} else if (condition2) {
    statement when condition1 is false
    and condition2 is true
} else {
    statement (otherwise)
}
```

- In this template, else if statement makes processing 3 branches.
- To add "`else if (condition i) { ··· }`", we can make processing multidirectional branches.

Example

```
if (diff < -10) {
  System.out.println("Underweight.");
} else if (diff <= 10) {
  System.out.println("Normal.");
} else {
  System.out.println("Overweight.");
}
```



Example program

Weight4.java

```
public class Weight4{
  public static void main(String[] args){
    double diff, height, std_weight, weight;

    height = 1.73;
    weight = 68.0;
    System.out.println("Height:" + height + "m, Weight:" + weight + "kg");

    std_weight = height * height * 22;
    System.out.println("Standard weight : " + std_weight + "kg");

    diff = weight - std_weight;
    if(diff < -10){
      System.out.println("Underweight");
    } else if(diff<=10) {
      System.out.println("Normal.");
    } else {
      System.out.println("Overweight.");
    }
  }
}
```

Multiway branch using switch statement

switch statement

```
switch (int expression) {
    case int expression 1:
        statement 1
        break;
    case int expression 2:
        statement 2
        break;
      ⋮
    default:
        statement n
}
```

- Switch statement takes an integer type and selects among a number of alternative `case` branches.
- The `default` case, which is optional, can be used to perform actions when none of the specified cases matches the int expression.
- The keyword `break` is optional. The break statement immediately ends the switch statement.

Example of `switch` statement

SampleSwitch.java

```
public class SampleSwitch {
  public static void main(String[] args) {
    int x = *;
    switch (x) {
      case 1:
        System.out.println("x is 1.");
        break;
      case 2:
        System.out.println("x is 2.");
        break;
      case 3:
        System.out.println("x is 3.");
        break;
      default:
        System.out.println("other number.");
    }
  }
}
```