# Decisions: if and if-else statements

Pieter Joubert

October 8, 2023

Decisions: if and if-else statements

## Table of Contents

Section 1

## Planning Decision Making Logic

## Making Decisions

- So far we have just looked at storing and manipulating data, with some organisation into methods and classes.
- To really start using the power of our computing devices we need to implement decision making in our programs.
- We have two tools we can use to help us with planning our decisions before we implement them: *Pseudocode* and *Flow-Charts*

## Pseudocode

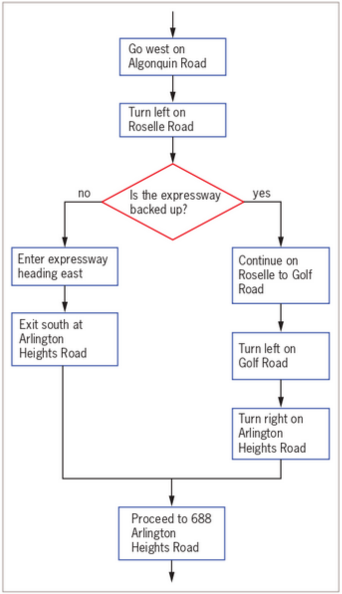
- Pseudocode, as the name implies, is a natural language approach to writing out an algorithm or plan in your code, before writing it in the final language, e.g. Java.

```
Get numbers from user.
Check if the numbers are bigger than zero and smaller
      than 100.
Add the numbers together.
Display the numbers back to the user.
```

## Flow-Charts

- Flow-Charts are diagrams that show the flow of a program.
- Flow-Charts use various symbols to indicate the different parts of program (e.g. assigning variables, making decisions, display output).
- Flow-Charts use arrows to indicate the flow of logic between these symbols.

## Flow-Charts: Example


Go west on Algonquin Road
Turn left on Roselle Road
Is the expressway backed up?
no
yes
Enter expressway heading east
Exit south at Arlington Heights Road
Continue on Roselle to Golf Road
Turn left on Golf Road
Turn right on Arlington Heights Road
Proceed to 688 Arlington Heights Road

Section 2

## if and if-else statements

## The if statement

- The simplest decision is the *if* statement, also known as the *single-alternative selection*.
- We test a condition in the *if* statement. If that condition is true we execute the code in the body of the *if* statement. If the condition is not true the execution of the code carries on after the body of the *if* statement.

```
if(<condition>) {
    <code to execute if true>
}
<code to execute after checking condition>
```

## The if statement: Example

```
if (age >= 18) {
    System.out.println("Can vote!");
}

System.out.println("Enter name")
```

## The if-else statement

- Very often we want to execute some code not only when a condition is true but also if it is false.
- In this case we use the $if - else$ statement. If the condition is true we execute the first block of code. If the condition is not true we execute the *else* block of code.

## The if-else statement: Example

```
if (age >= 18) {
    System.out.println("Can vote!");
} else {
    System.out.println("Cannot vote!");
}

System.out.println("Enter name")
```

Section 3

## Using multiple statements in if and if-else statements

## Multiple Actions

- Very often we want to execute multiple statements if a certain condition is true.
- And multiple statements if a condition is false.
- In either case we need to ensure that all the statements we want to execute are within the same *if* or *else* block.

```
if (health > 0) {
    health = health - damge;
    System.out.println(health);
} else {
    isDead = true;
    System.out.println("You have died");
}
```

Section 4

## Nesting if and if-else statements

## Nested *if* statements

- We can execute almost any type of code within an *if* statement.
- This includes other *if* statements.
- This can be a useful technique to split up the decisions you need to make into smaller units.

```
if (health > 0) {
    health = health - damge;
    System.out.println(health);
} else {
    if(resurrectionStone == true) {
        health = 1;
        isDead = false;
    } else {
        isDead = true;
        System.out.println("You have died");
    }
}
```

## Nested *if* statements - example

- Let's code up an example of using nested if statements to play Rock-Paper-Scissors
- We accept some input from the user (R,P,S)
- We will then use a nested if statement to determine what is the best response to the user's move and then print out that move.

Section 5

## Using Logical AND and OR operators

## Multiple conditions in one if statement

- In Java we can have multiple conditions in one if statement.
- Instead of checking if only ONE condition is true, we can check if condition1 and condition2 are both true, or if at least one is true.
- We can do this with as many conditions as we want.
- The AND operator is written using two ampersand characters: &&
- The AND operator is written using two pipe characters: ||
- Let's update the nested if example to use the AND operator instead.

## AND example

```
if (health > 0 && resurrectionStone == true) {
    health = health - damage;
    System.out.println(health);
} else {
    isDead = true;
    System.out.println("You have died");
}
```

- Note that this logic is not exactly the same as the previous example.
- In this case you could die even if your health is above zero if you don't have the resurrection stone.
- Be very careful with using the AND and OR operators. Always double check that your logic is making sense.

## OR example

```java
if (health > 0 || resurrectionStone == true) {
    health = health - damage;
    System.out.println(health);
} else {
    isDead = true;
    System.out.println("You have died");
}
```

- Note that once again this logic is not exactly the same as the nested if example.
- In this case you wouldn't die if you don't have the resurrection stone but you could get into a situation where you end up with negative health.

Section 6

## Lecture summary

## Lecture summary

- Planning Decision Making Logic
- if and if-else statements
- Using multiple if and if-else statements
- Nesting if and if-else statements
- Using Logical AND and OR operators

## Questions

**Thank you! Questions?**