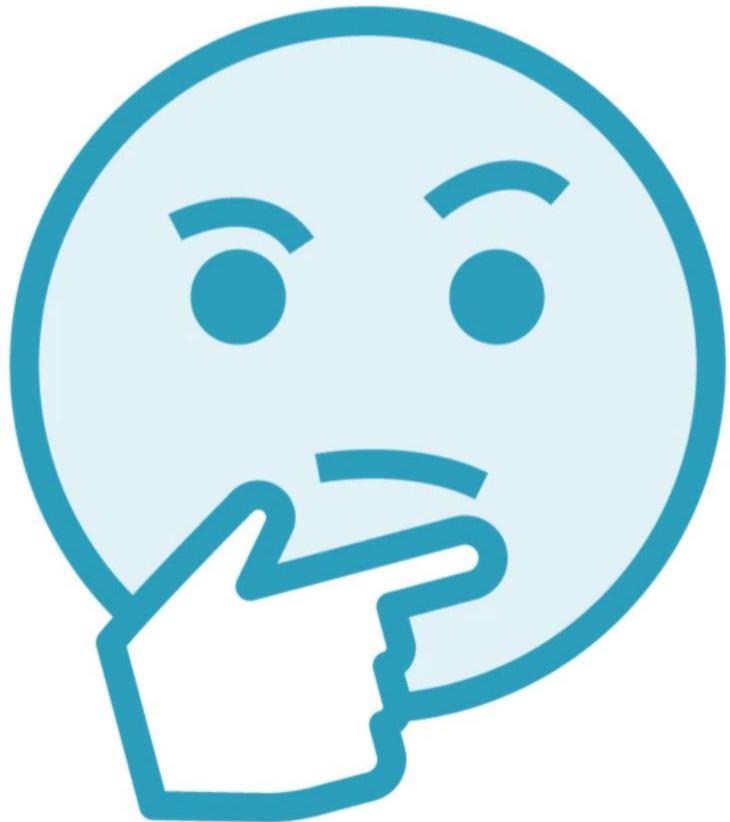




A multi-faceted language for the Java platform

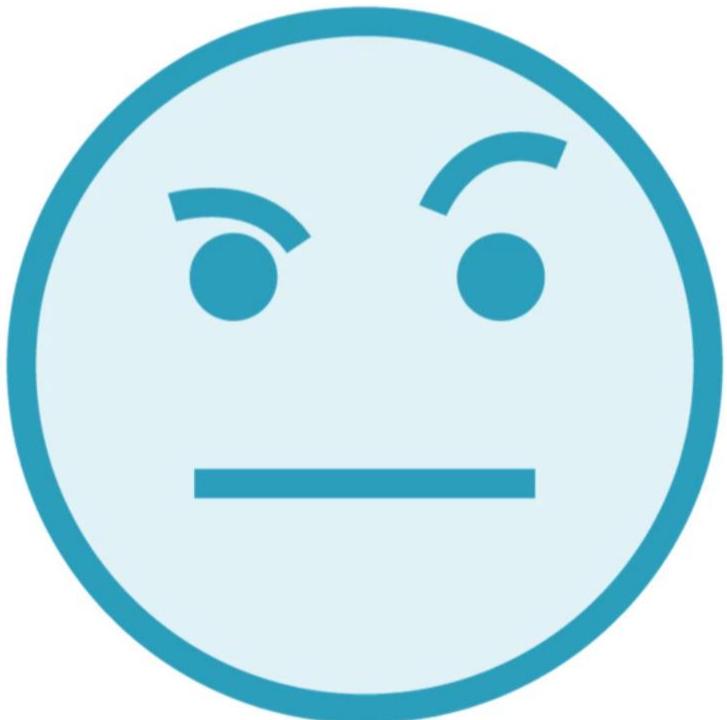
Apache Groovy is a powerful, **optionally typed** and **dynamic language**, with **static-typing** and **static compilation** capabilities, for the Java platform aimed at improving developer productivity thanks to a concise, **familiar and easy to learn syntax**. It integrates smoothly with any Java program, and immediately delivers to your application powerful features, including scripting capabilities, **Domain-Specific Language** authoring, runtime and compile-time **meta-programming** and **functional programming**.

Why Learn Groovy?



Groovy is an extension to Java
Write concise code
Automate recurring tasks

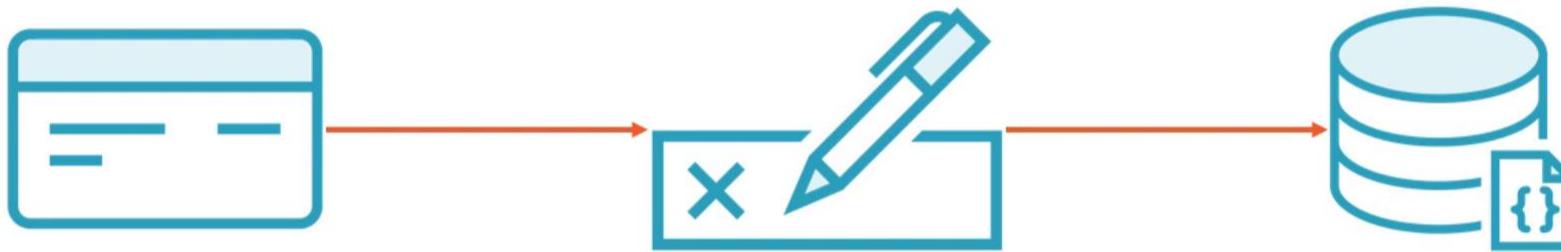
Who Is Using Groovy?



Netflix
LinkedIn
Walmart
IBM
Google
Oracle

Java	Groovy
It is developed on JDK and is run on JVM	It is compiled to JVM Byte code and It is compatible with Java platform
It is used as programming and object oriented Language	It is used as both programming and scripting Language
In Java default access modifier package	In Groovy default access modifier public
In Java , you need to provide getters and setters method for fields, especially if you are following Java Beans naming convention	In Groovy , getters and setters are automatically generated for class members
In Java semicolons are compulsory	In Groovy semicolons are optional
In Java only Java.lang.* package is imported by default	In Groovy commonly used packages are imported by default
Java has primitive data types and wrapper classes to perform boxing and unboxing implicitly or explicitly	In Groovy everything is object and uses only object hence no concept of autoboxing or unboxing
Java has a requirement of the main method inside a class to run the program	Groovy does not require any main method or entry point of a method to run the class or any program

Understanding the Business Problem



Payment card industry (**PCI**) compliance is mandated by credit card companies to help ensure the security of credit card transactions in the payments industry

Companies need to mask the credit card numbers while displaying and mask in database logs

Understanding the Business Problem



Understanding the Business Problem



- Scan the directory containing the log files**
- Filter files with .log extension**
- Scan for credit cards**
- Mask card data except the last four digits**
- Write the run time to a database**
- Enhance the utility to process XML and JSON files**

Groovy Editors



SlickEdit

Textmate

IntelliJ

Vim

UltraEdit

IDE integration

Many IDEs and text editors support the Groovy programming language.

Editor	Syntax highlighting	Code completion	Refactoring
Groovy Eclipse Plugin	Yes	Yes	Yes
IntelliJ IDEA	Yes	Yes	Yes
Netbeans	Yes	Yes	Yes
Groovy Emacs Modes	Yes	No	No
TextMate	Yes	No	No
vim	Yes	No	No
UltraEdit	Yes	No	No
SlickEdit	Yes	No	No
EditRocket	Yes	No	No
vsCode	Yes	No	Yes

<https://groovy-lang.org/ides.html>

Environment Setup

Groovy 4 is compatible with any Java release Greater than version 8

JDK Download:

<https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html>

Groovy Download:

<https://groovy.apache.org/download.html>

<https://groovy.jfrog.io/ui/native/dist-release-local/groovy-zips/apache-groovy-sdk-4.0.1.zip>

<https://groovy.jfrog.io/ui/native/dist-release-local/groovy-zips/apache-groovy-sdk-4.0.6.zip>

IntelliJ Download:

<https://www.jetbrains.com/idea/download/#section=windows>

Groovy is an optionally typed, dynamic language for the Java platform with features inspired by Python, Ruby, and Smalltalk, making them available to Java developers using a Java-like syntax.

Value Proposition of Groovy



Created as a companion to Java

Increase the productivity of Java developers

Feature-rich and Java-friendly compared to Scala, Kotlin, and Ceylon

Can be called from a Java program

Offers scripting capabilities

A complete object-oriented language

Ability to process XML, SQL, and JSON

Dynamic programming

Groovy Comments



A single line comment starts with //

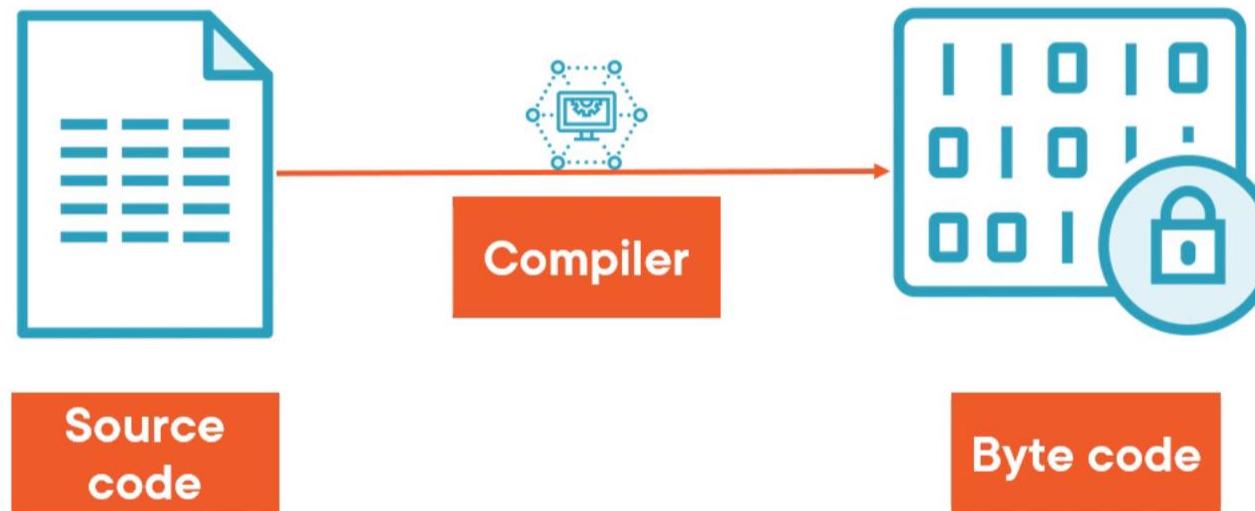
A multi-line comment starts with /* and ends with */

Compiled vs. Interpreted

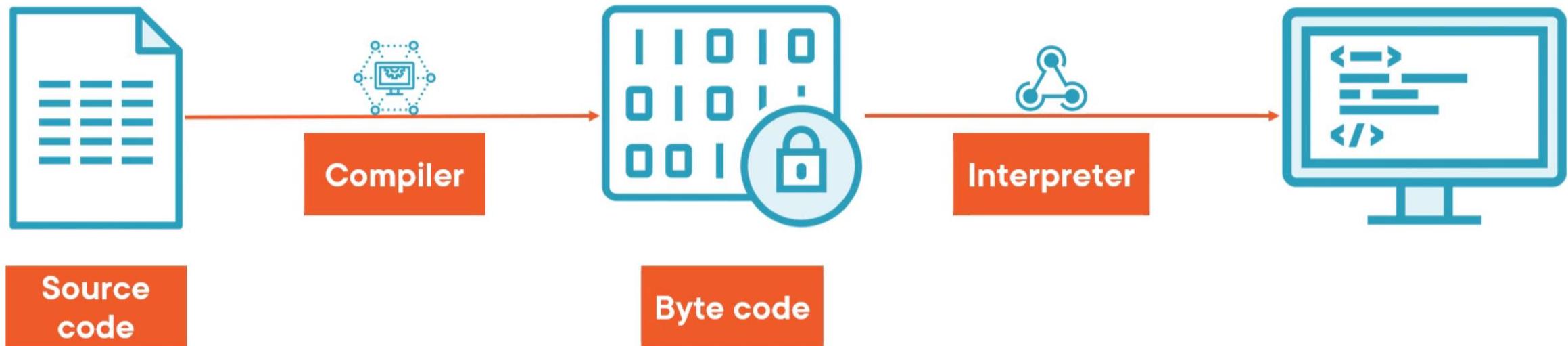


Source
code

Compiled vs. Interpreted



Compiled vs. Interpreted



Compiled vs. Interpreted



First groovy class & the compiler

File Name: Example.groovy

```
class Example {  
    static void main(String[] args) {  
        // Using a simple println statement to print output to the console  
        println('Hello World')  
    }  
}
```

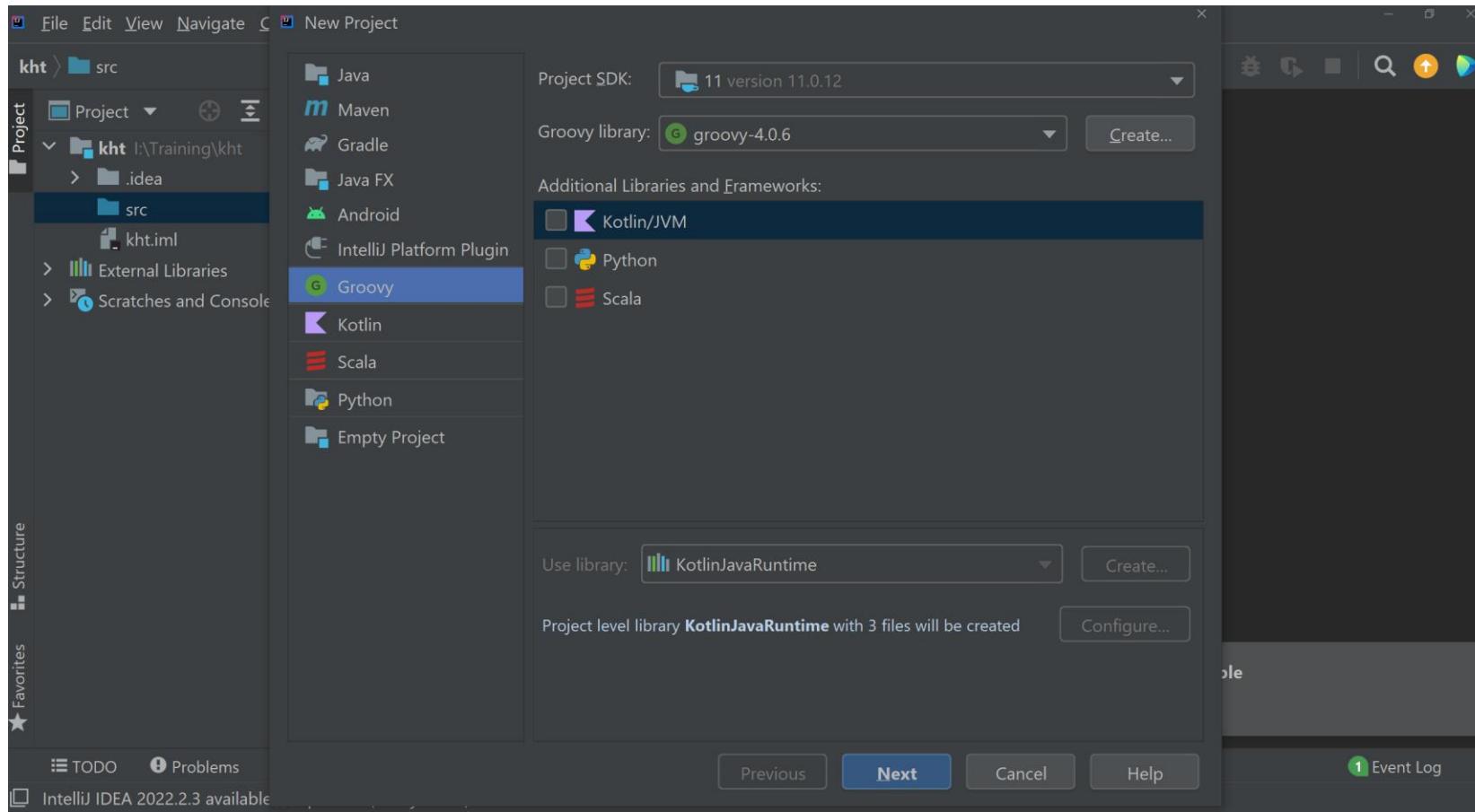
Groovyc compiler:

```
groovyc Example.groovy -> Example.class
```

Execute the class/groovy file:

```
groovy Example.groovy / groovy Example (with compiled file)
```

Create Groovy Project from IntelliJ



Groovy utilities

Step 1: Open Command Prompt

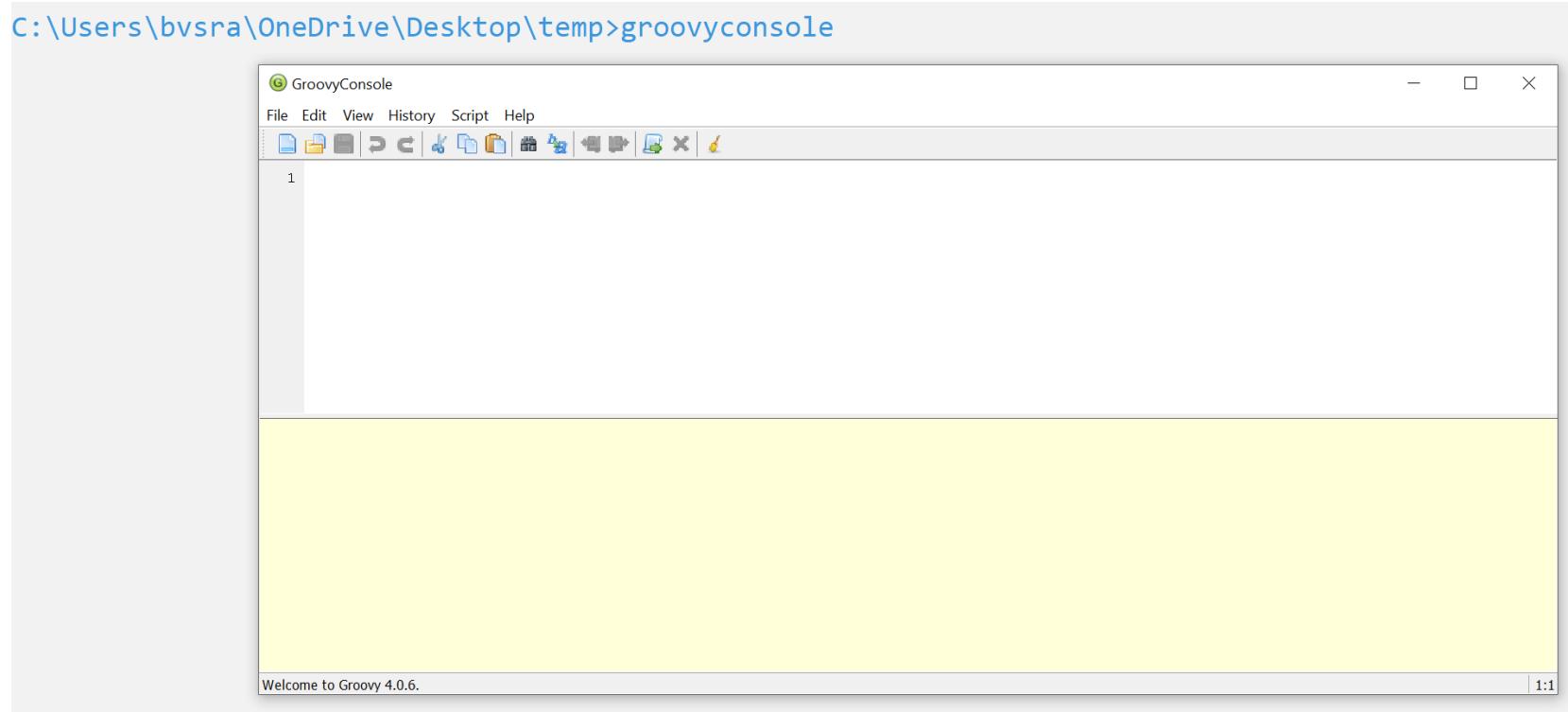
Step 2: execute **groovysh** utility in command prompt

```
C:\Users\bvsra\OneDrive\Desktop\temp>groovysh
Groovy Shell (4.0.6, JVM: 11.0.12)
Type ':help' or ':h' for help.
-----
groovy:000> 1
==> 1
groovy:000> i = 10
==> 10
```

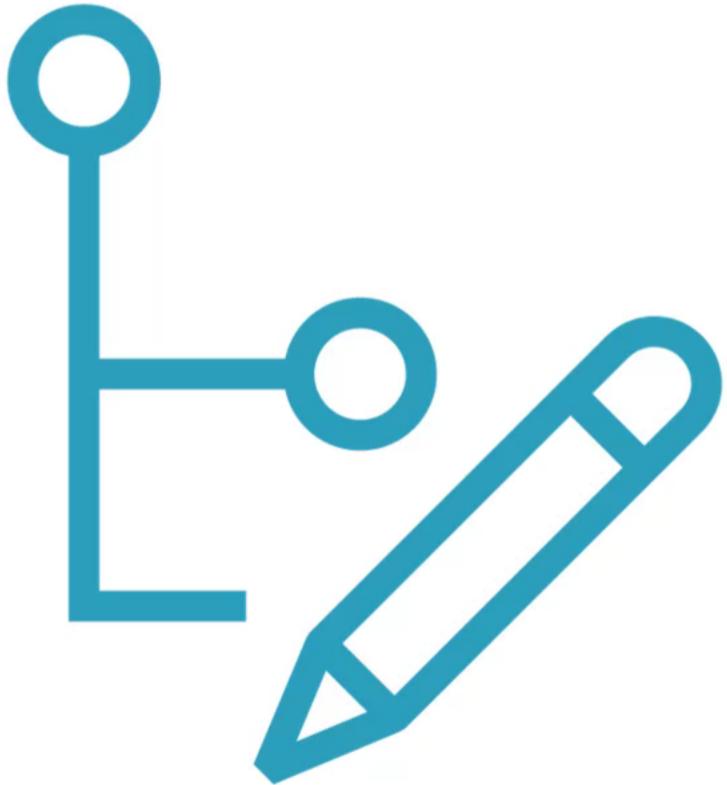
Groovy utilities

Step 1: Open Command Prompt

Step 2: execute **groovyconsole** utility in command prompt



Groovy Datatypes



byte, short, int, and long

- Whole numbers

float, double

- Decimal point numbers

char, string

- Characters and string

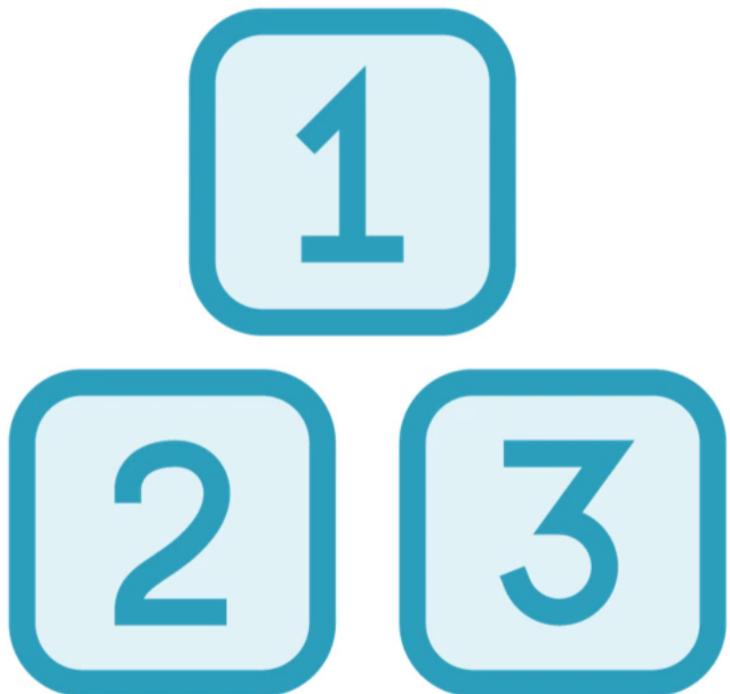
boolean

- Logical values

Groovy Datatypes Capacity

Datatype	Description	Capacity
byte	8-bit signed integer	-128 to +127
short	16-bit signed integer	-32768 to +32767
int	32-bit signed integer	- 2^{31} to $2^{31}-1$
long	64-bit signed integer	- 2^{63} to $2^{63}-1$
float	Single-precision 32-bit floating point	approximately $\pm 3.40282347E+38F$
double	Double-precision 64-bit floating point	approximately $\pm 1.79769313486231570E+308$
char	16-bit Unicode character	'\u0000' (or 0) to '\uffff' (or 65,535 inclusive)
boolean	Boolean value	true and false

Groovy Arithmetic Operations



Addition

Subtraction

Multiplication

Division

Modulus

Increment

Decrement

Operator Precedence

$5 + 5 / 5 * 5$

$5 + 1 * 5$

$5 + 5$

10

Operator Precedence

$5 + 5 / (5 * 5)$

Operator Precedence

$5 + 5 / (5 * 5)$

$5 + 1 / 25$

$5 + 0.2$

5.2

Boolean Values



A true value is represented by integer 1

A false value is represented by integer 0

Supports AND, OR, and NOT

Logical AND

Operand1	Operand2	Result
		
		
		
		

Logical OR

Operand1	Operand2	Result
		
		
		
		

Logical NOT

Operand	Result
	
	

Relational Operators



`== -> object equality`

`!= -> object inequality`

Java Strings

Single quoted string

Cannot interpolate variables

Double quoted string

Can interpolate variables

Dynamic Typing



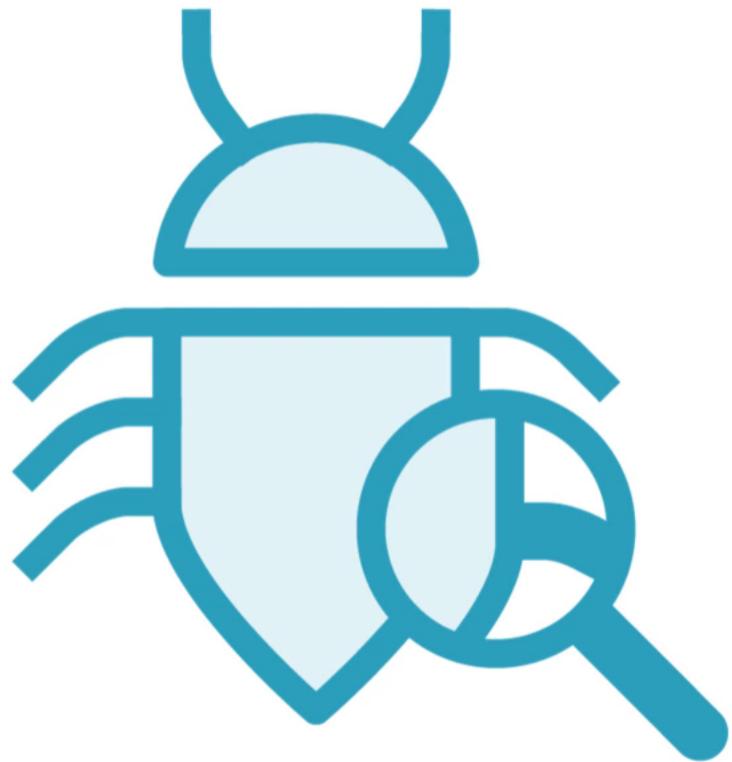
Types are inferred based on context
Not strictly enforced during compile-time

Dynamic Typing Benefits



- Less verbose code**
- Explicit type casting not required**
- Dynamic testing**
- Effective debugging**

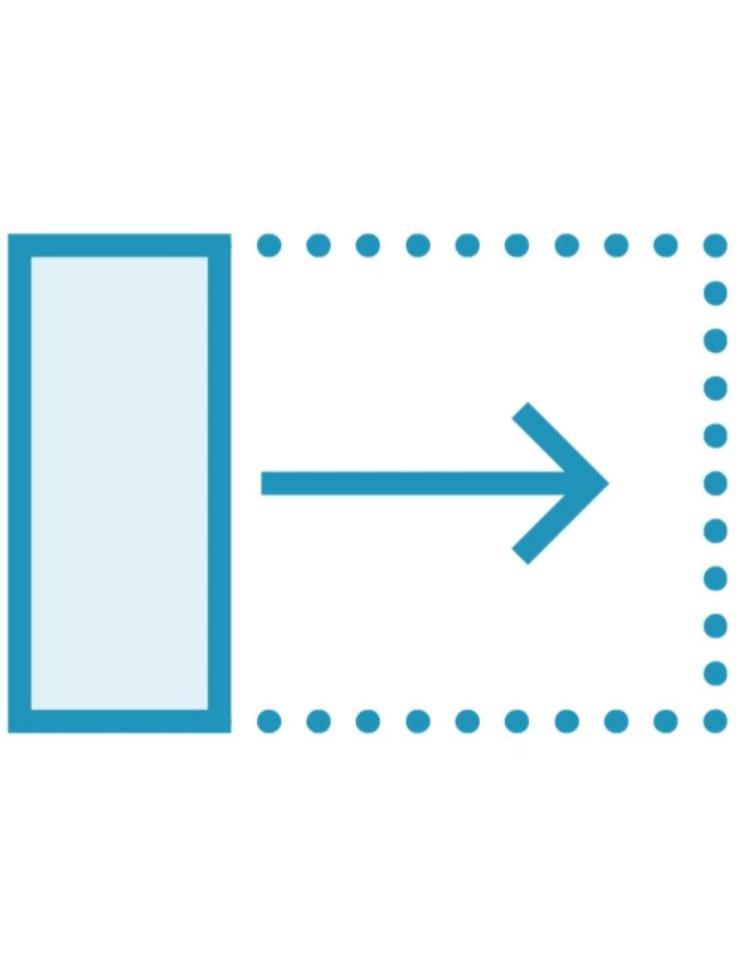
Unit Testing



Assertion helps validate an expression

Built-in feature in Groovy

Groovy Range



Not a good fit for storing random collections

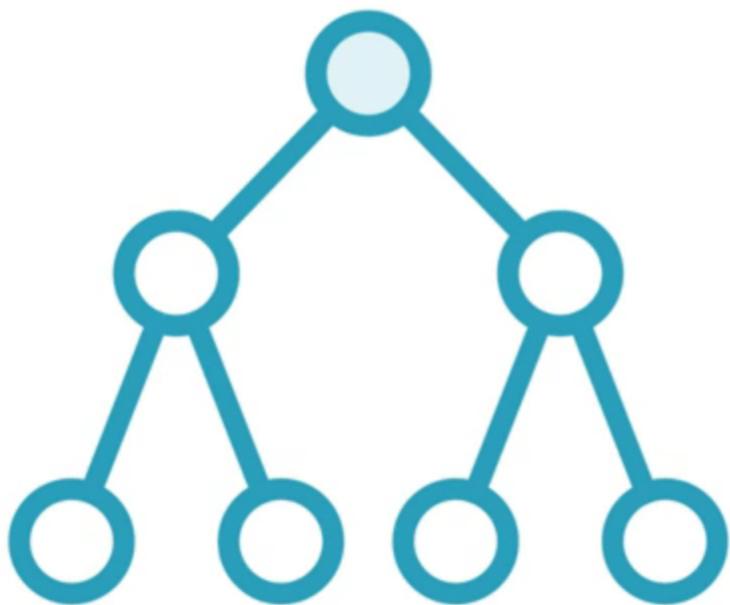
Groovy Map



Stores elements as key-value pair

Branching Logic in Groovy

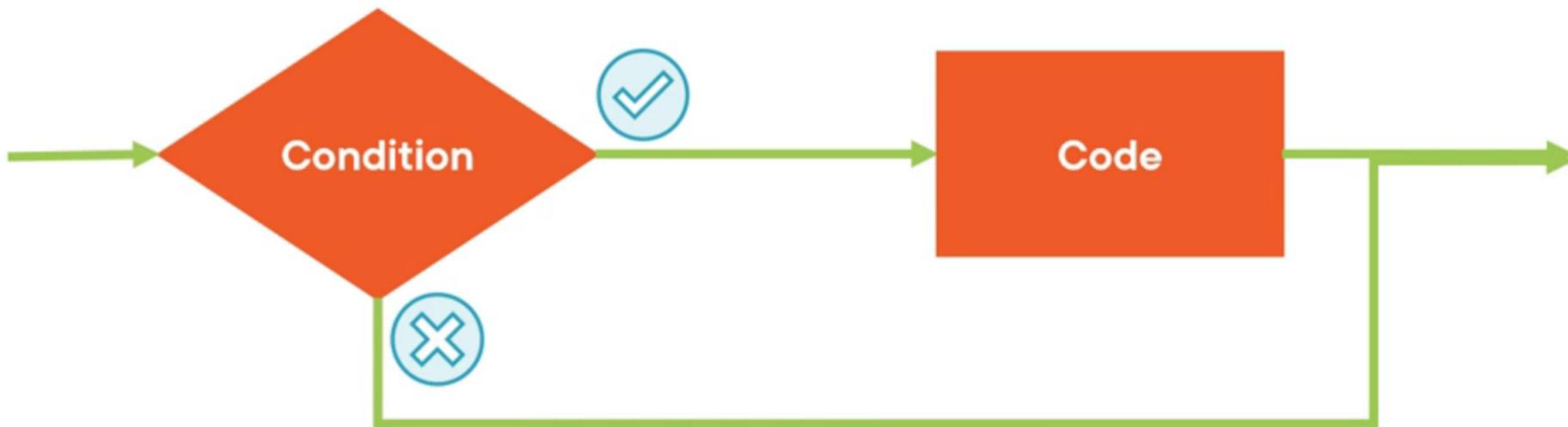
Branching Logic in Groovy



Executing code based on certain condition?

- Use “if” statement

Branching Logic in Groovy



Business Requirements



Execute a code block when the list has four elements

Business Requirements

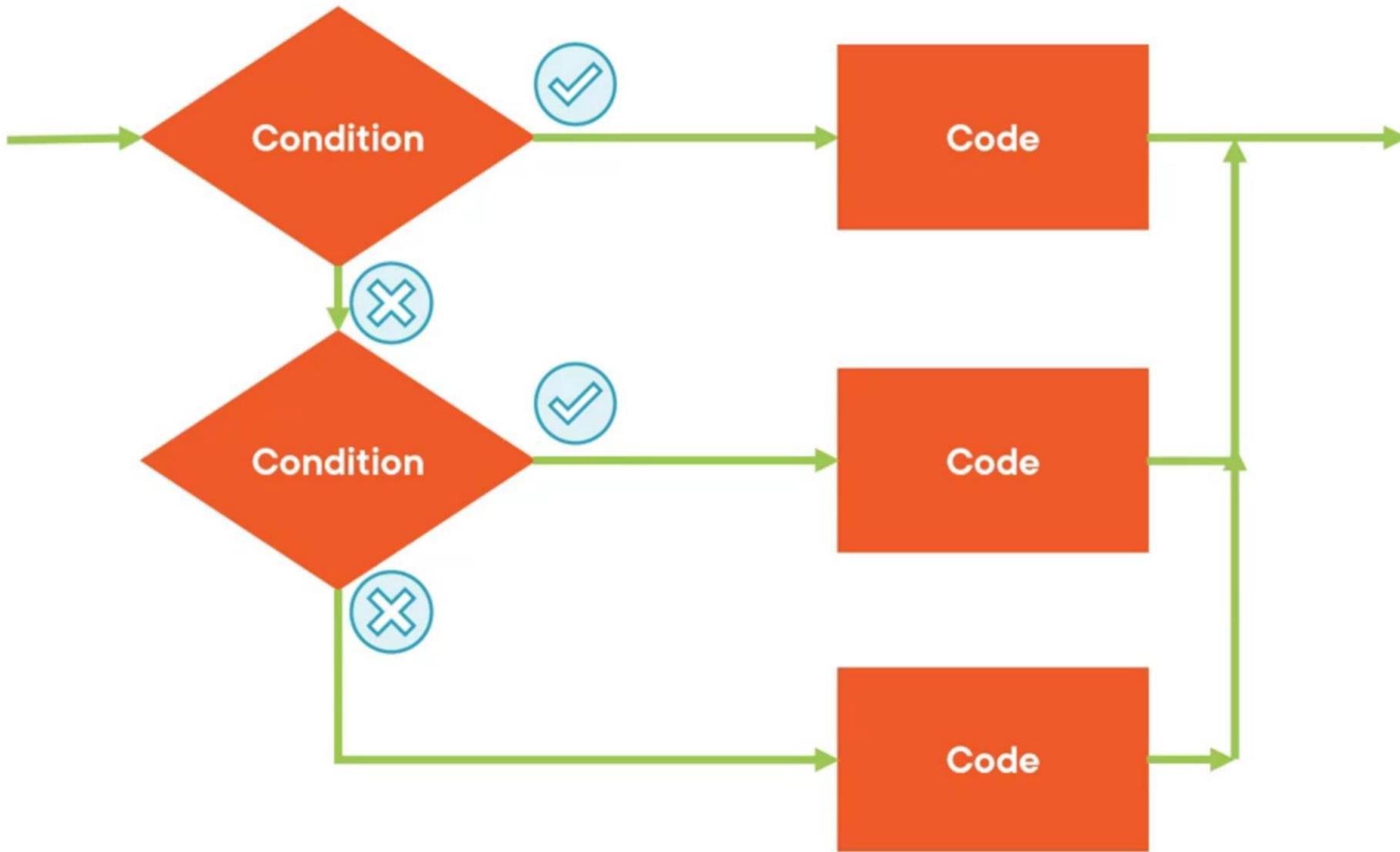


Execute a code block when the list has four elements

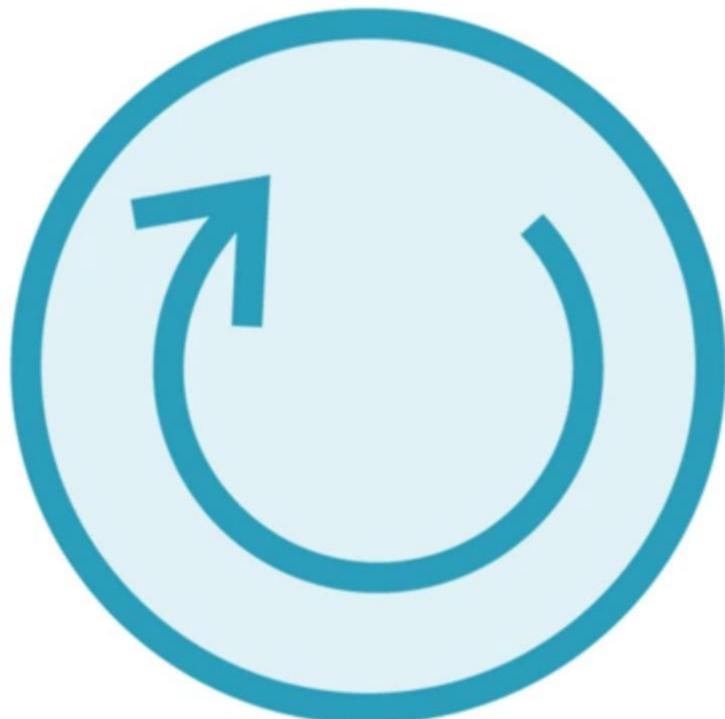
Execute a different code block when the list has five elements

Print a default message for other scenarios

Branching Logic in Groovy



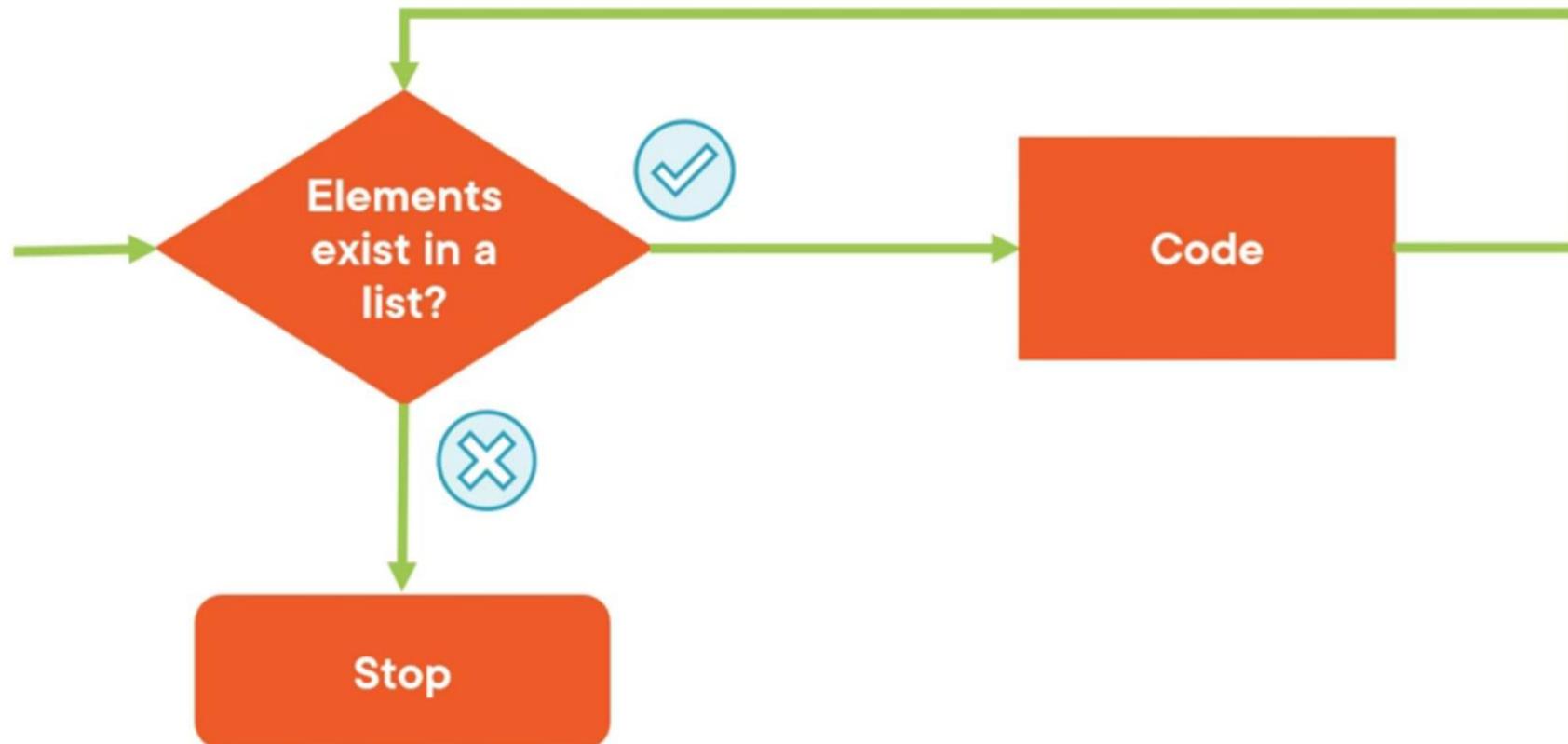
Looping Logic in Groovy



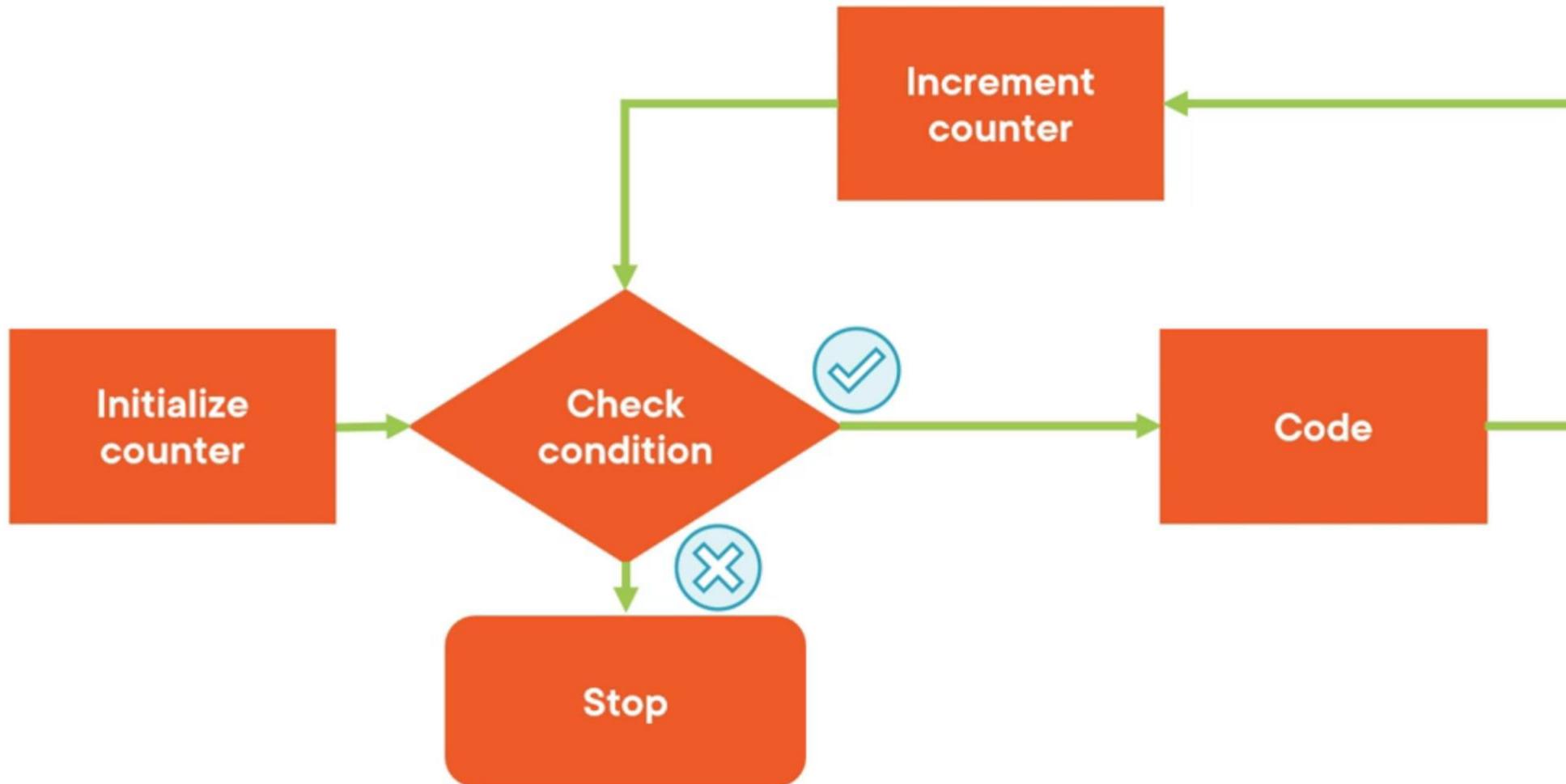
Need to iterate over all the elements in a collection?

- Use “for” loop

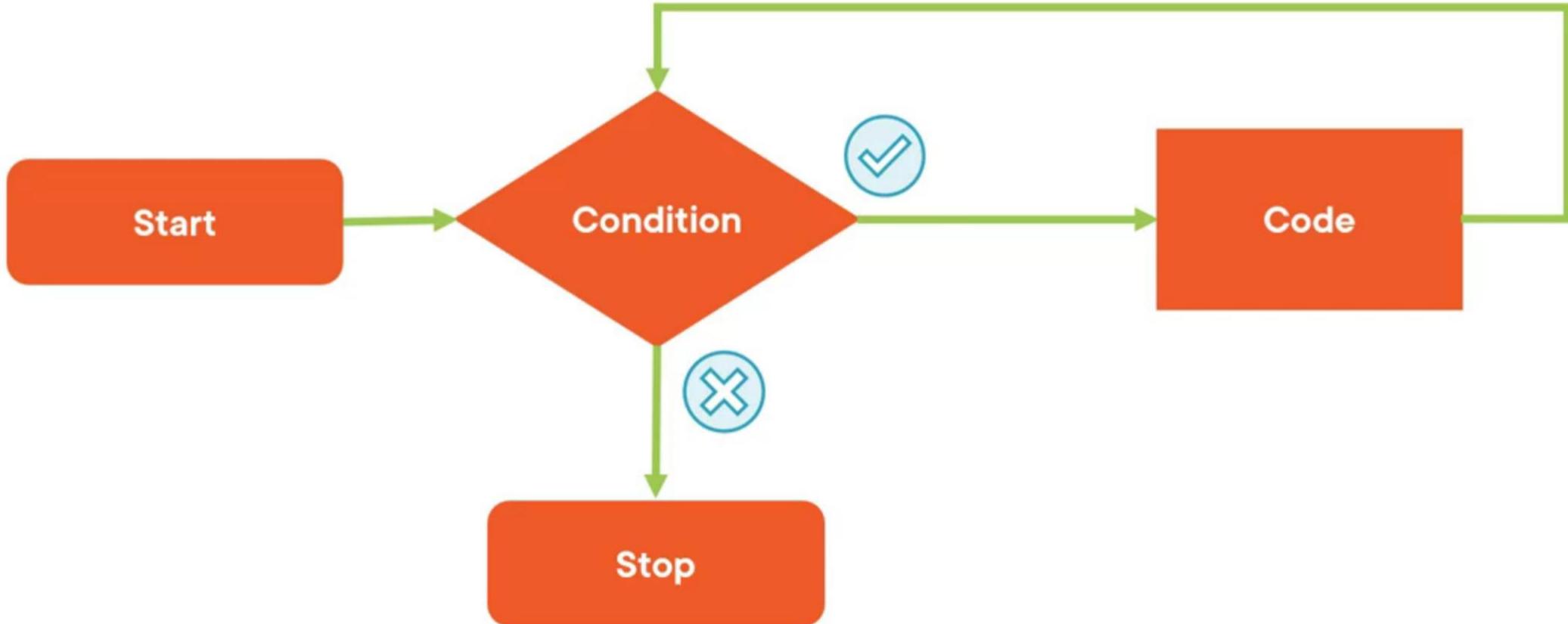
for Loop in Groovy



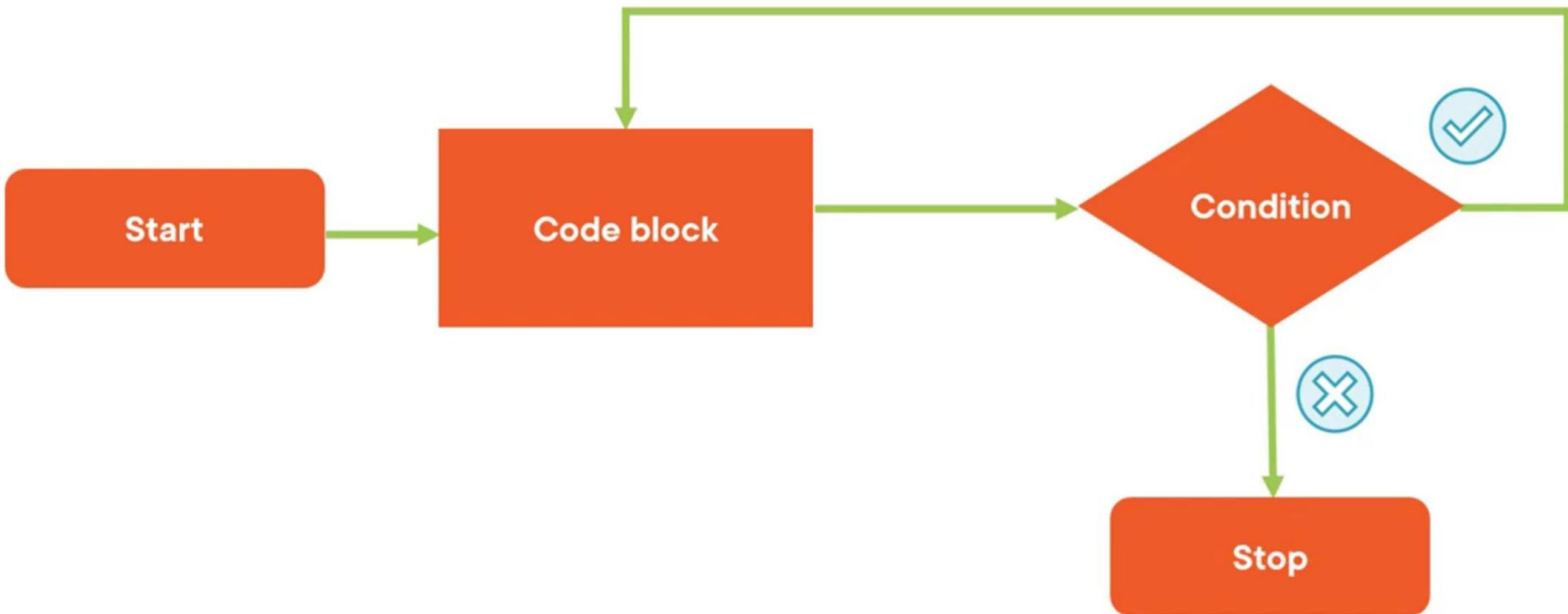
for Loop in Groovy



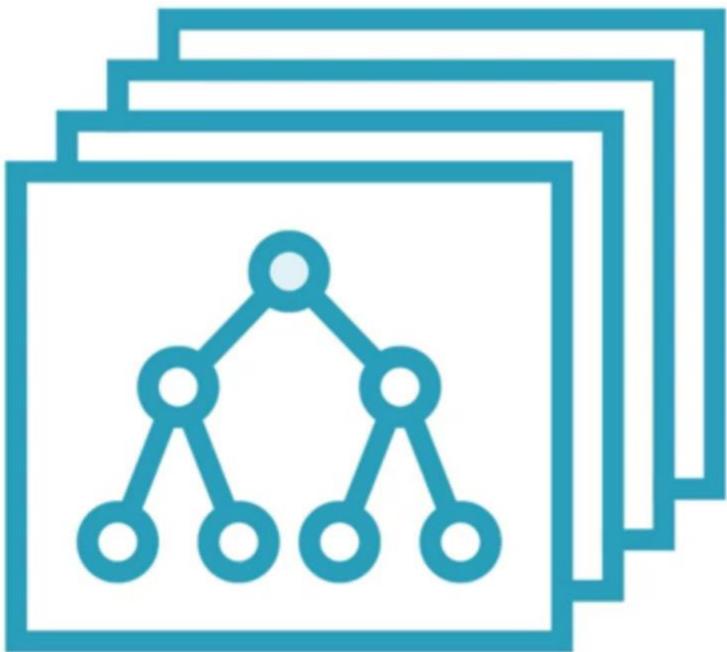
while Loop in Groovy



do/while Loop in Groovy

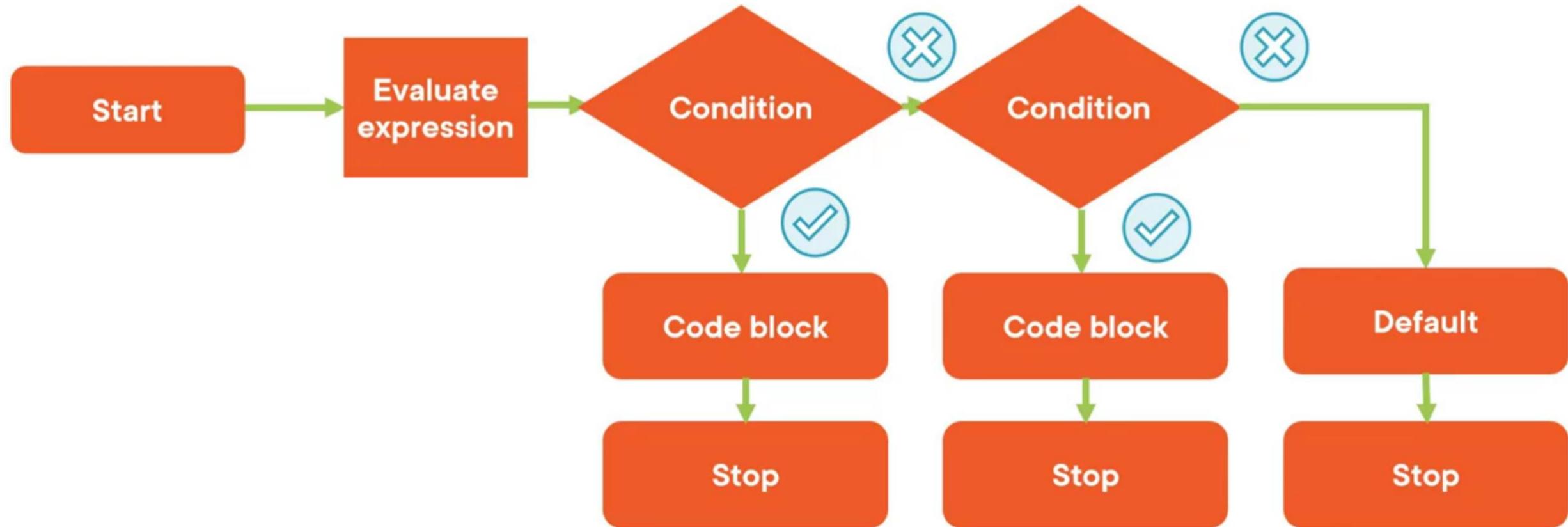


Switching Logic in Groovy



Reduce code clutter

Switch Statements in Groovy



Business Scenario



- Prompt the user to enter a credit card type**
- Print a message for all valid card types**
- Print a warning message for any other string**

Business Scenario



A new element has been added to the list
Skip the execution while processing this new element
Use Groovy “continue” statement

Business Scenario

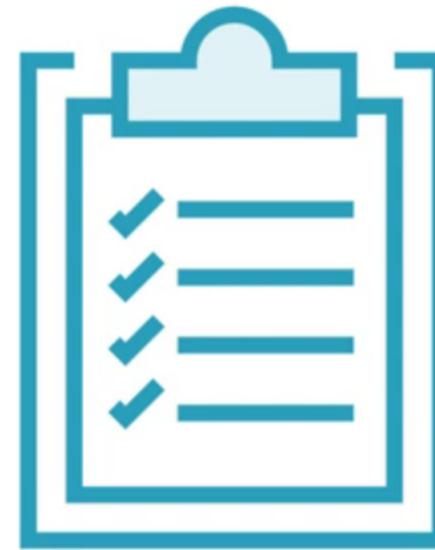


A new element has been added to the list
Stop the execution while processing this new element
Use Groovy “break” statement

Iterating Using Groovy Built-in Methods

Exception Handling





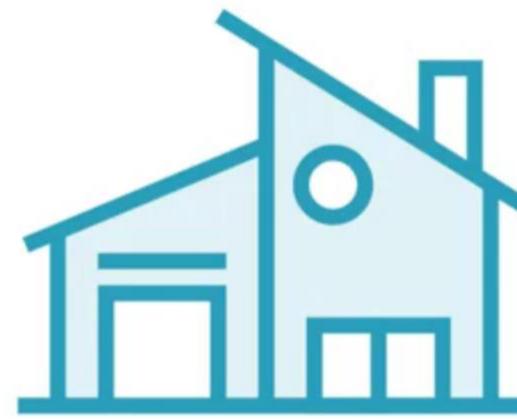
Checked Exception



Groovy maintains a checklist of rules and validates against it during a program execution

A deviation from this rule is a checked exception

Checked at compile time



Runtime Exception



Not handled during compile time

- **NullPointerException**
- **ArrayIndexOutOfBoundsException**

Throwing an Exception



Program cautions against unforeseen circumstances



Error Handling

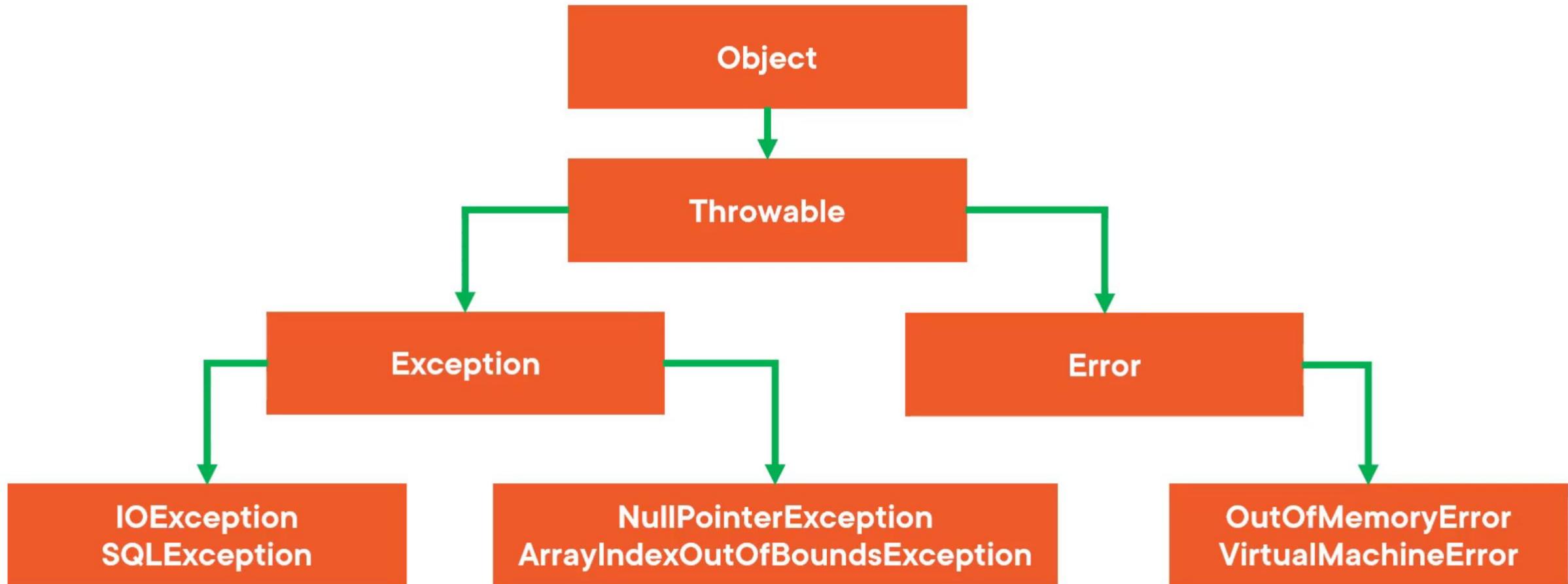


An exception is recoverable

An error is irrecoverable

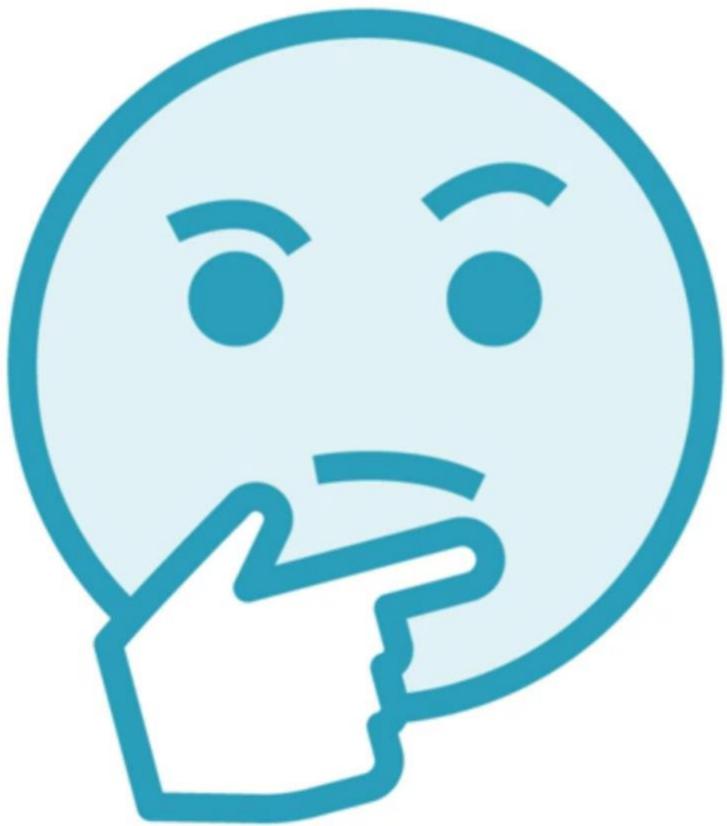
- **OutOfMemoryError**

Exception Hierarchy



Exception Handling Using try/catch Block

Exception Handling



Impractical to add catch blocks for all potential exceptions

Use the parent Exception class in the catch block

Performing Cleanup after Execution

Code Cleanup



Closing file handles

Closing database connection

Unclosed resources adversely affect the performance

Throwing an Exception

Business Requirement



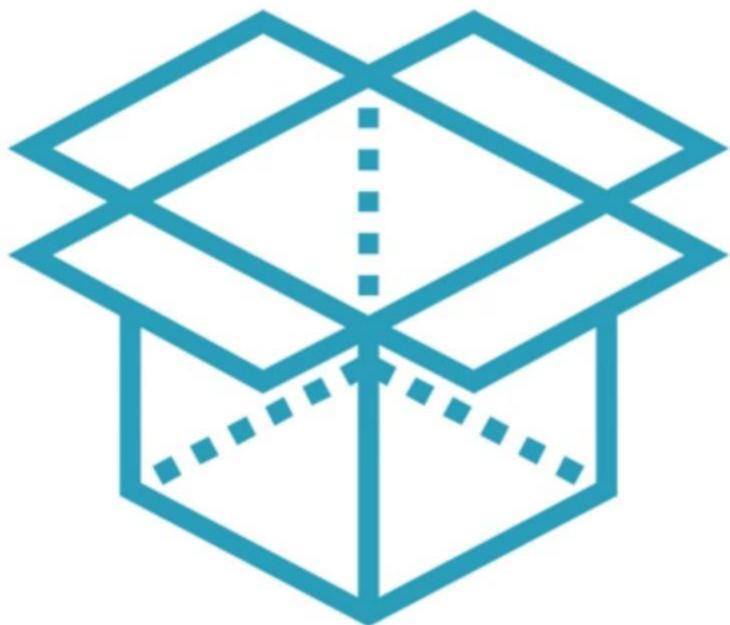
Process credit card containing 14 or more digits only

Throw an InvalidCardLengthException for cards containing fewer than 14 digits

Introducing Closures

A closure is an open, anonymous block of code that can take arguments, return a value, and be assigned to a variable. A closure may reference variables declared in its surrounding scope.

Closures



Can be treated as objects

Can be passed as method parameters

Helps write concise code

Used as an iterator, callbacks, higher-order functions, and builders

Passing Parameters and Returning Values in Closures

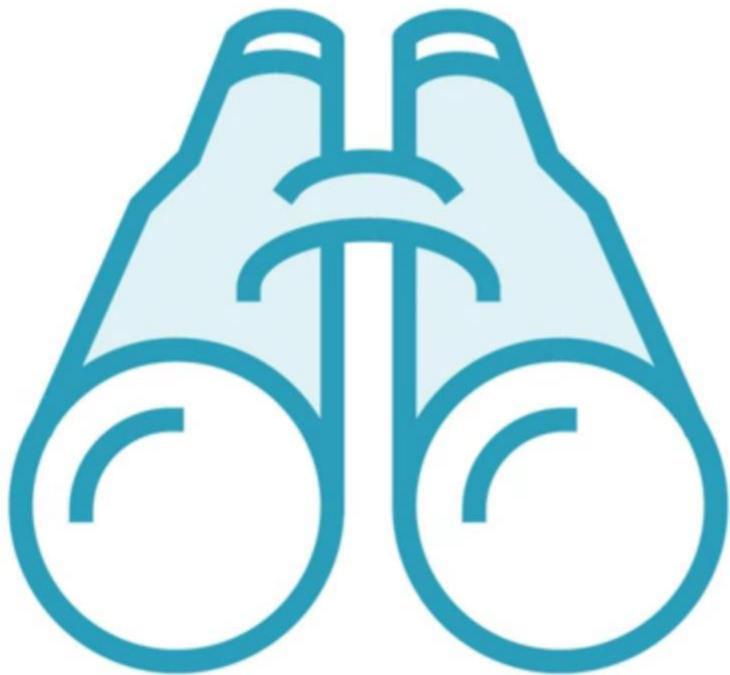
```
def sample_closure = { <params> -> <statements> }
```

Using Closures in Methods and Collections

A closure is an open, anonymous block of code that can take arguments, return a value, and be assigned to a variable. A closure may reference variables declared in its surrounding scope.

Using Closure Scope and Delegates

Closure Scope



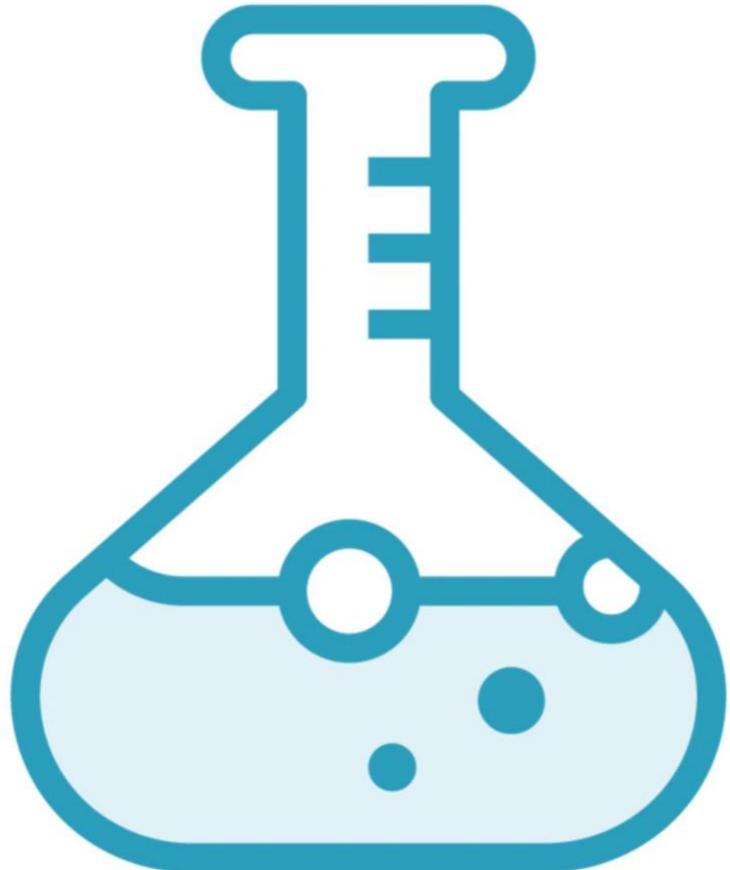
Visibility of a closure

Three levels of scope

- **this**
 - Corresponds to the enclosing class
- **Owner**
 - Corresponds to the enclosing object where it is defined
- **Delegate**
 - Corresponds to a third-party object where method calls are resolved
 - By default, the delegate is set to owner

Groovy Closure Advanced Features

Currying



A closure accepting three parameters

Invoked multiple times

One of the parameters is a constant

Create another closure based on first closure by passing a default constant parameter

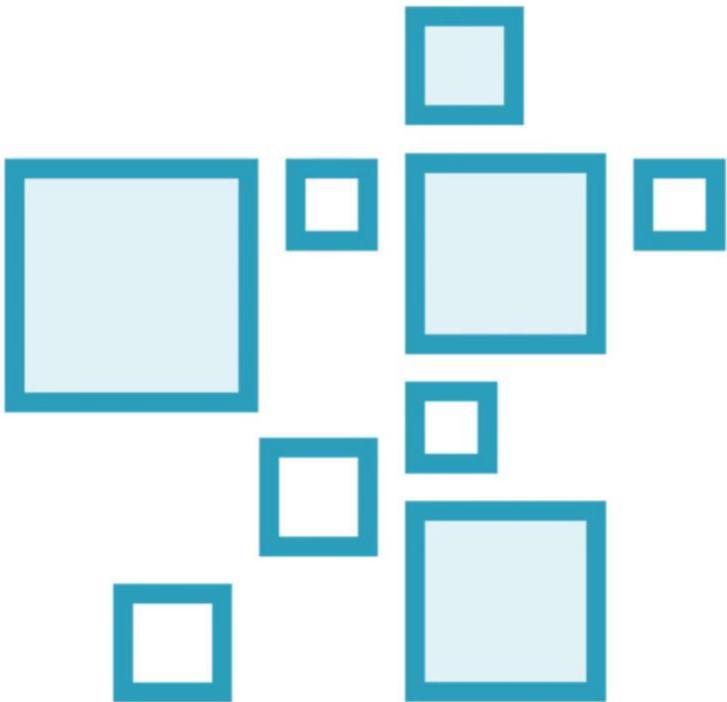
Memoization



Used in closures performing resource intense computation tasks

Cache the results using memoize function

Composition



Chain multiple calls by combining two or more functions

Create a closure by composing two or more closures

Used in Groovy DSLs

Working with Files in Groovy

Business Requirements



- Fetch the directory name from the user**
- Open the directory**
- Read the files**
- Open the files with .log extension**

<https://groovy-lang.org/gdk.html>

Working with Regular Expressions

A regular expression is a pattern that is used to find substrings in text. Groovy supports regular expressions natively using the ~"regex" expression. The text enclosed within the quotations represent the expression for comparison.

https://docs.groovy-lang.org/latest/html/documentation/#_regular_expression_operators

Regular Expressions in Groovy



Not a comprehensive reference on regular expressions

Regular Expression Fundamentals

Business Requirements



Read the input file line by line

**Test if the line contains the word “amex” or
“visa”**

Print a statement to the console

Write the same message to the output file

Business Requirements



Removed the card type prefixes from the input file

Replaced each card occurrence with the text “card” followed by a number

Business Requirements



Read the input file line by line

**Test if the line contains the word “amex” or
“visa”**

Extract the credit card number

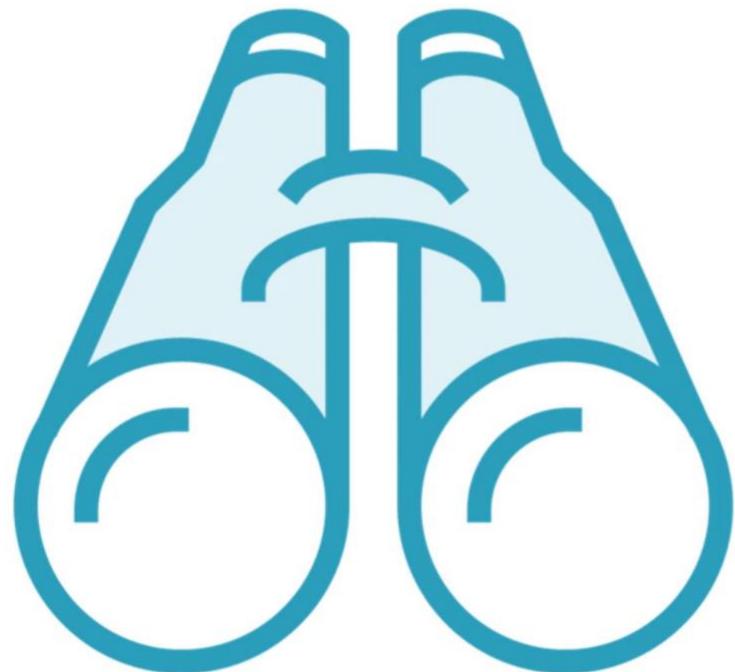
**Print the extracted number and the
corresponding line number**

Business Requirements



Search for a credit card in each line
Replace it with a star except for the last four digits

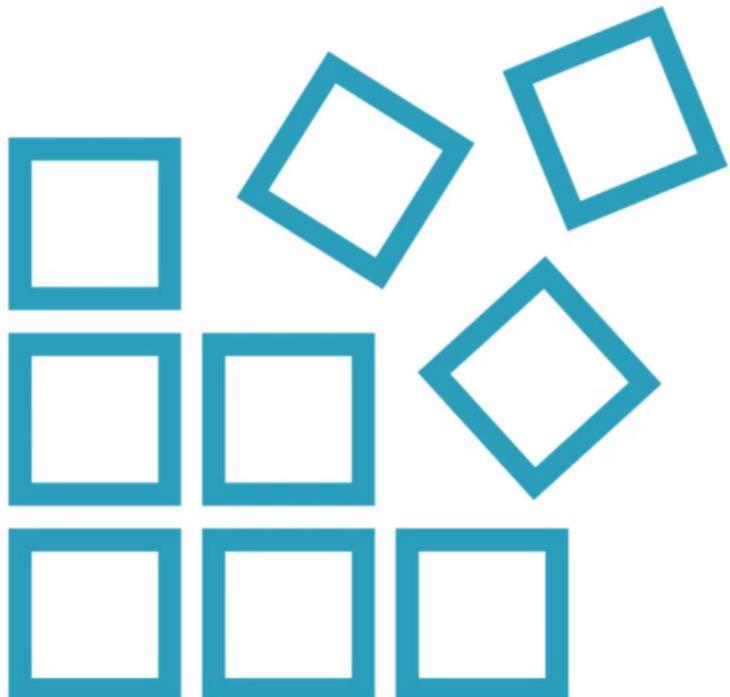
Lookahead Expression



Assertion used to determine the success or failure of the pattern ahead

(?= <expression>)

Current Logic

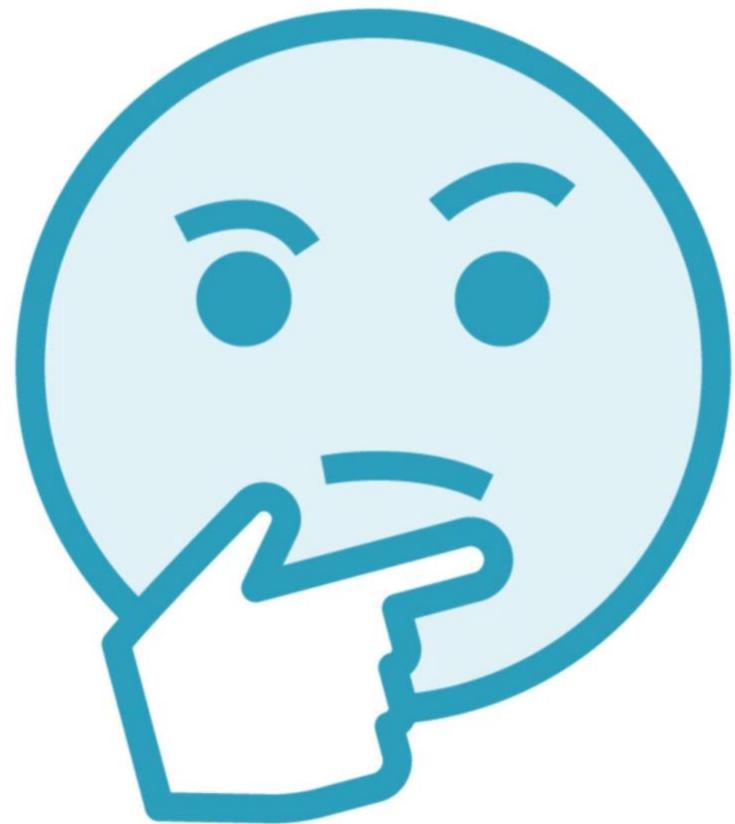


Used a while loop and read the input file

Extracted the credit card numbers except for the last four digits

Computed the number of stars to be used for masking dynamically

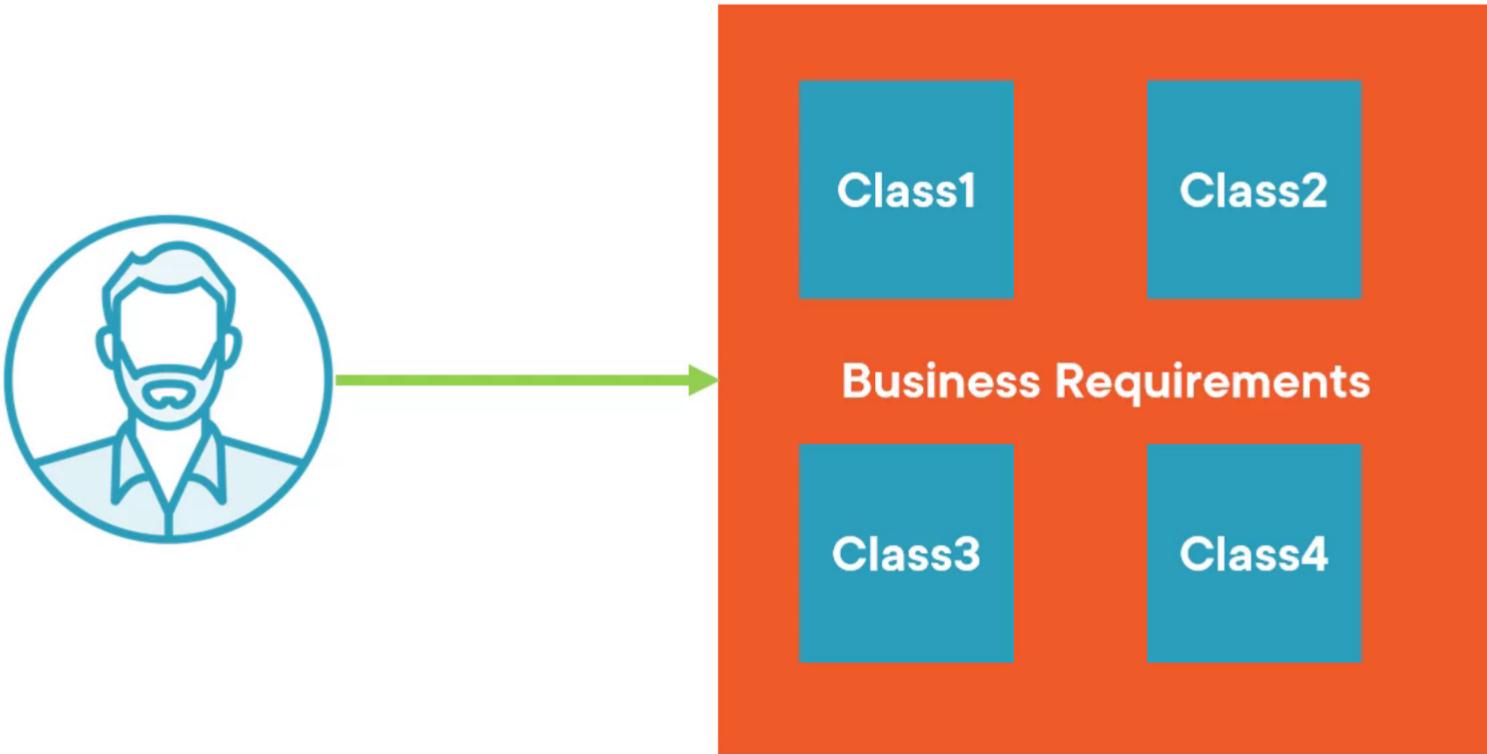
Pending Logic

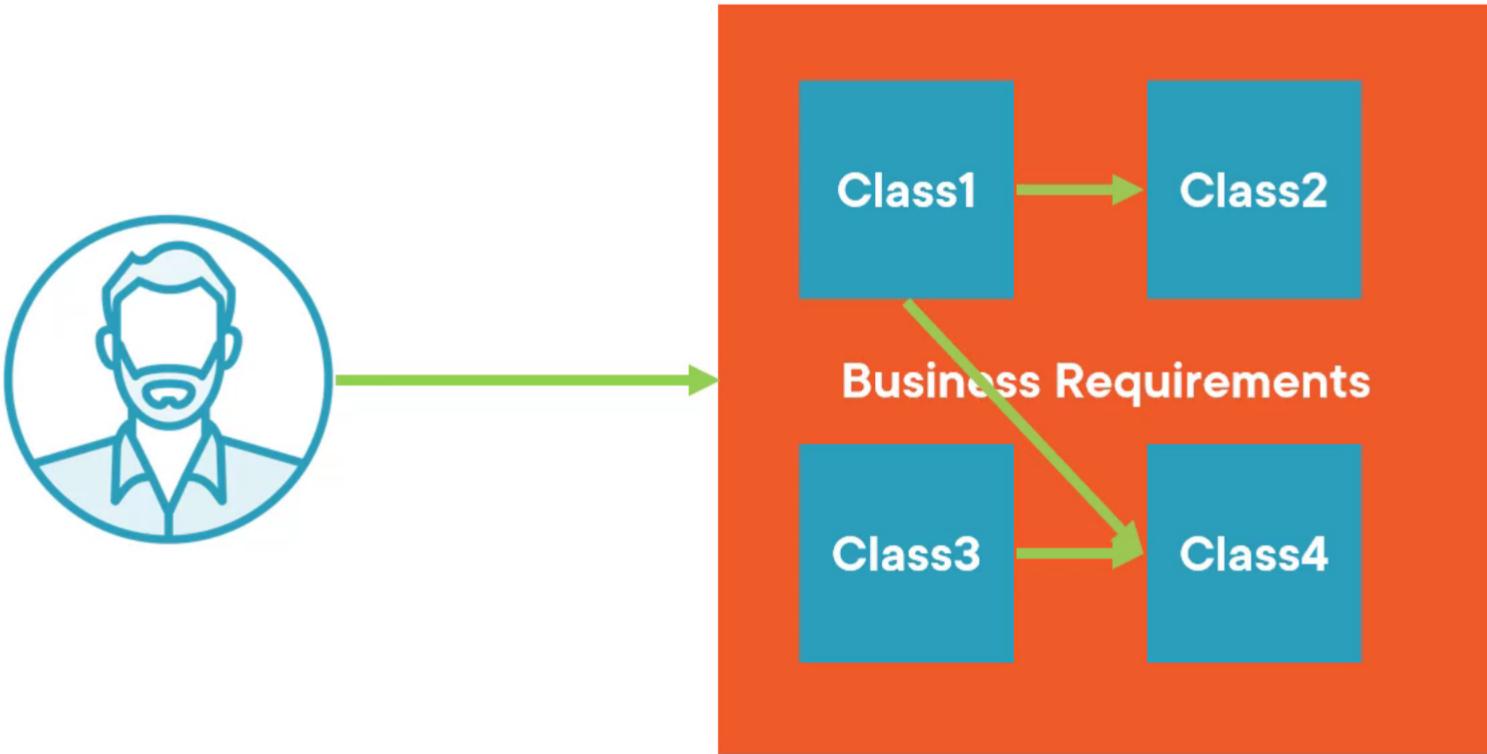


How to update the input file directly?

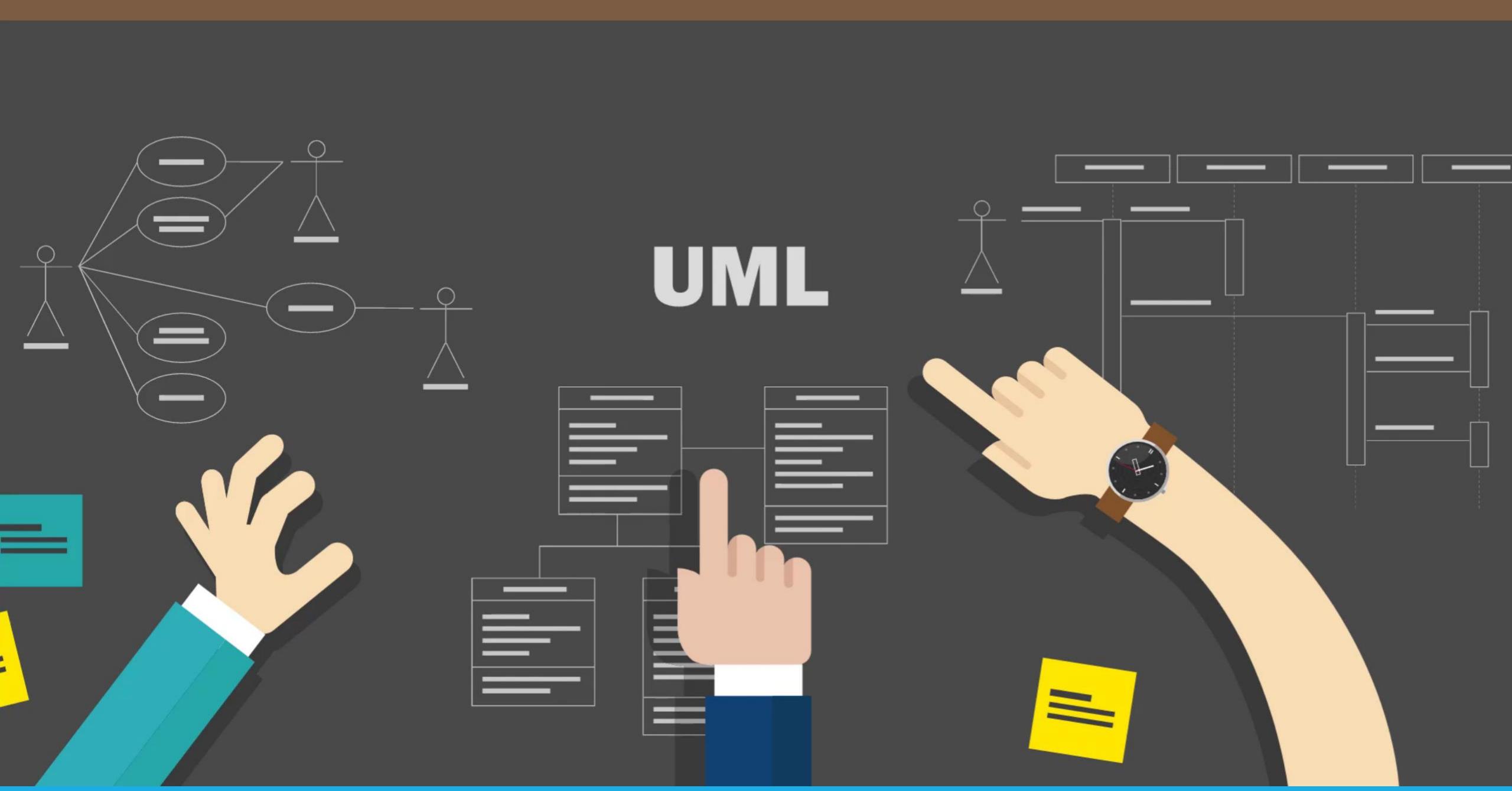
Object-oriented Programming

Defining Classes and Methods
Constructor overloading and Methods
Working with Inheritance
Working with Interface
Using Traits in Groovy
Using Groovy Beans





UML



Access Modifiers



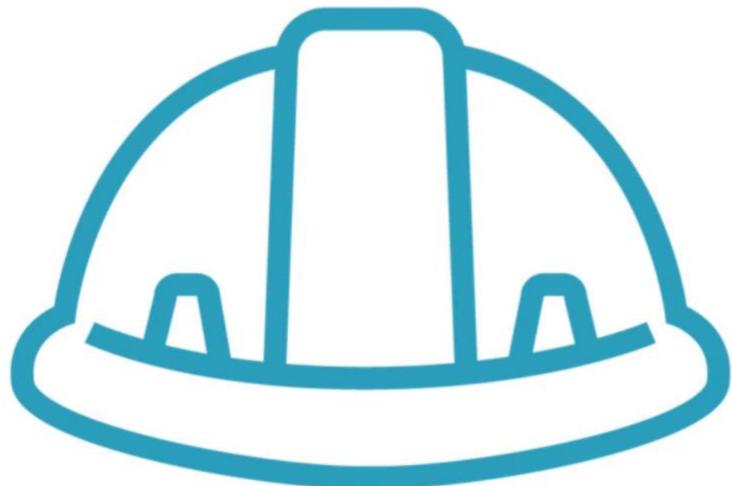
Controls the scope

Private access restrict the access outside the class

Protected access allows access to other classes in the same package

Public access has no restrictions

Constructors



Used to create an instance of a class

Groovy offers a default no argument constructor

The constructor's name must be same as the class name and public

Constructor Overloading and Methods

Properties and Methods



Properties control and reflect the inner working of a class

Methods are used to manipulate the properties

Properties must be private to prevent external access

Multiple Inheritance in Groovy



Groovy does not support multiple inheritance
Interfaces to the rescue

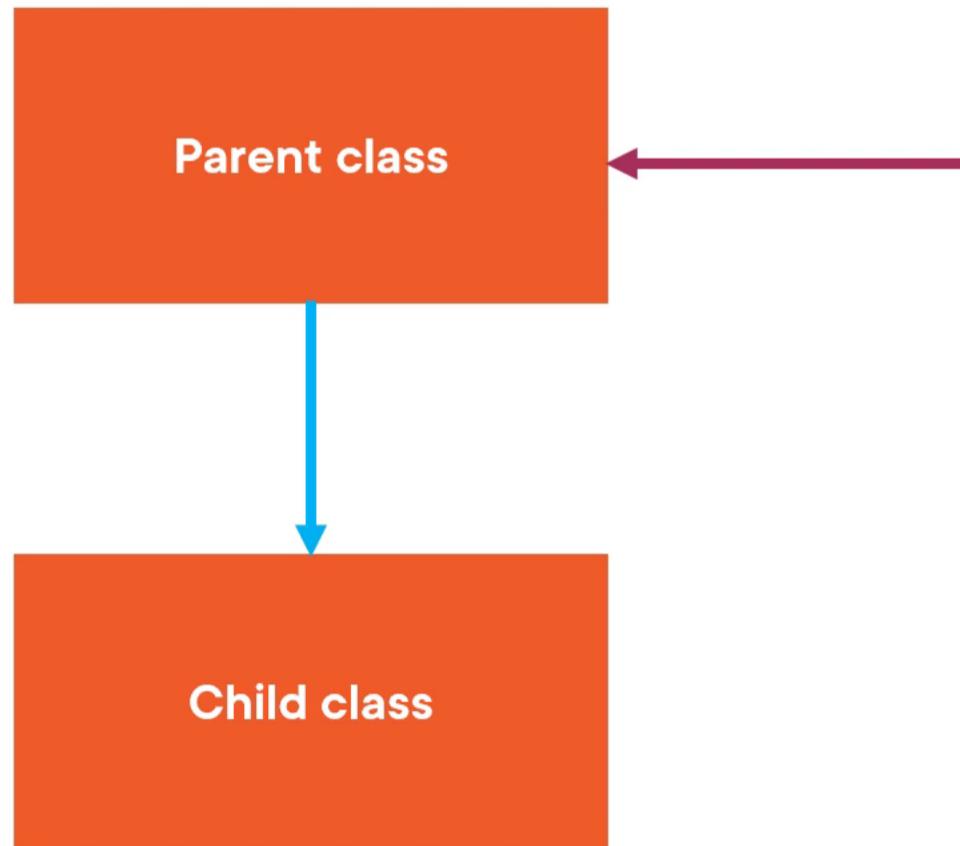
Interface in Groovy



All methods declared in an interface must be abstract

They carry only the method signature and no implementation

Call Hierarchy



Default Methods



A **default implementation** is provided by the interface



Java Bean

Holds private properties with getter and setter methods

Used to transport data from data tier to application tier

Business Requirements



Keep track of the card names and the number of digits processed by the tool

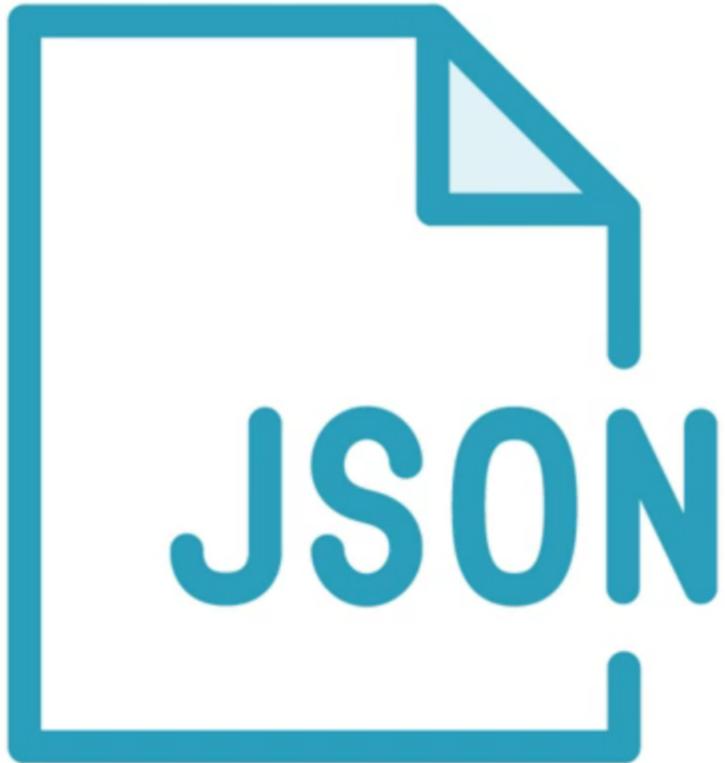
Processing XML and JSON in Groovy

Business Requirements



Print the card collection in xml format

JSON



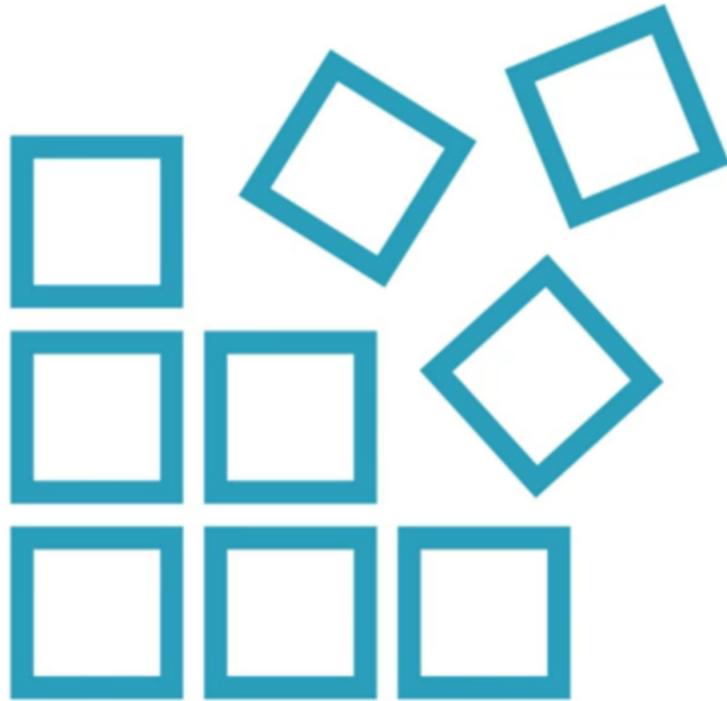
Common message format used in cross-tier communication

Payload Parsing



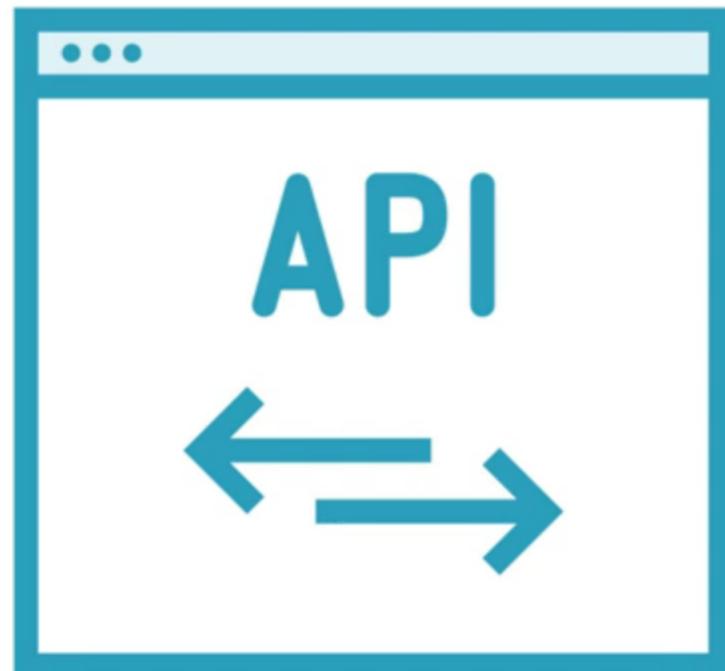
Parsing xml or json payload is relatively a simple process in Groovy

XML Parsing



**XmlSlurper lets you traverse the xml tree
similar to xpath expression**

REST



Common protocol used for fetching data from a server

GET and POST are commonly used

Stateless protocol

Metaprogramming

Metaprogramming



Code that generates code
Works one level above the code

你好

Hi

SELAM

안녕

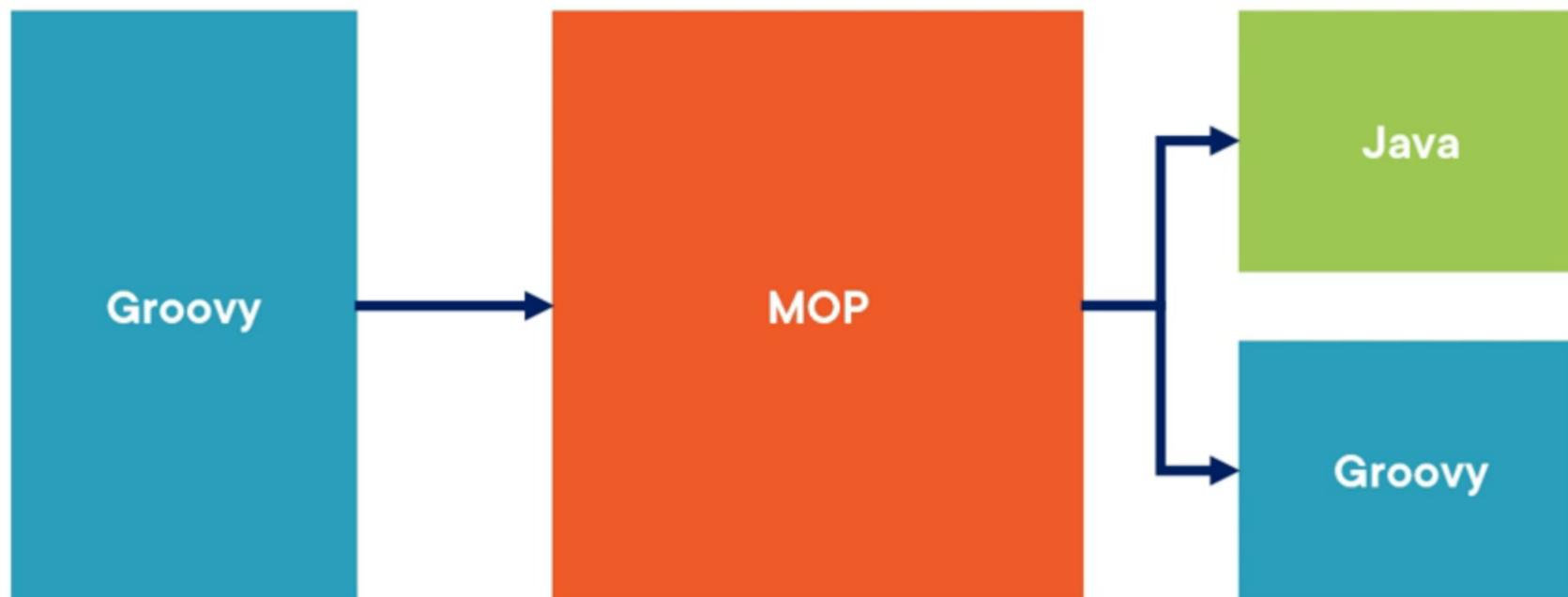
Hola

Salut



Translator





Method Hooks



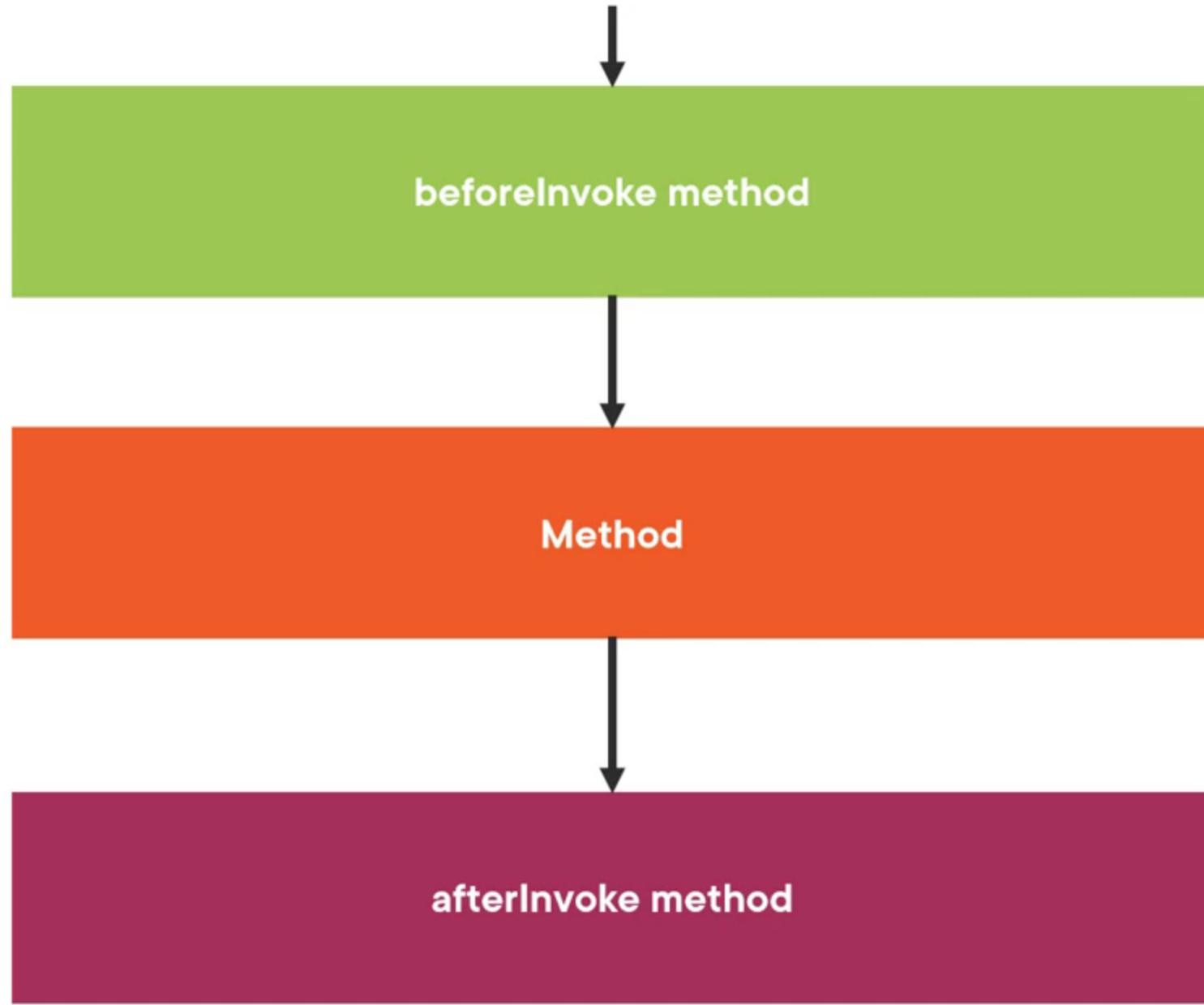
Need not to be static across instances
May change during the lifetime of an object

Aspect-oriented Programming

ProxyMetaClass



Used to achieve interception capabilities



Summary

<https://docs.groovy-lang.org/latest/html/documentation/core-syntax.html>

Groovy datatypes

Groovy operators and operator precedence

Value proposition of dynamic typing and its implementation

Assertion

Differences between inclusive and exclusive Range

Differences between Groovy Array and List

Add, remove, and search elements in Groovy Map