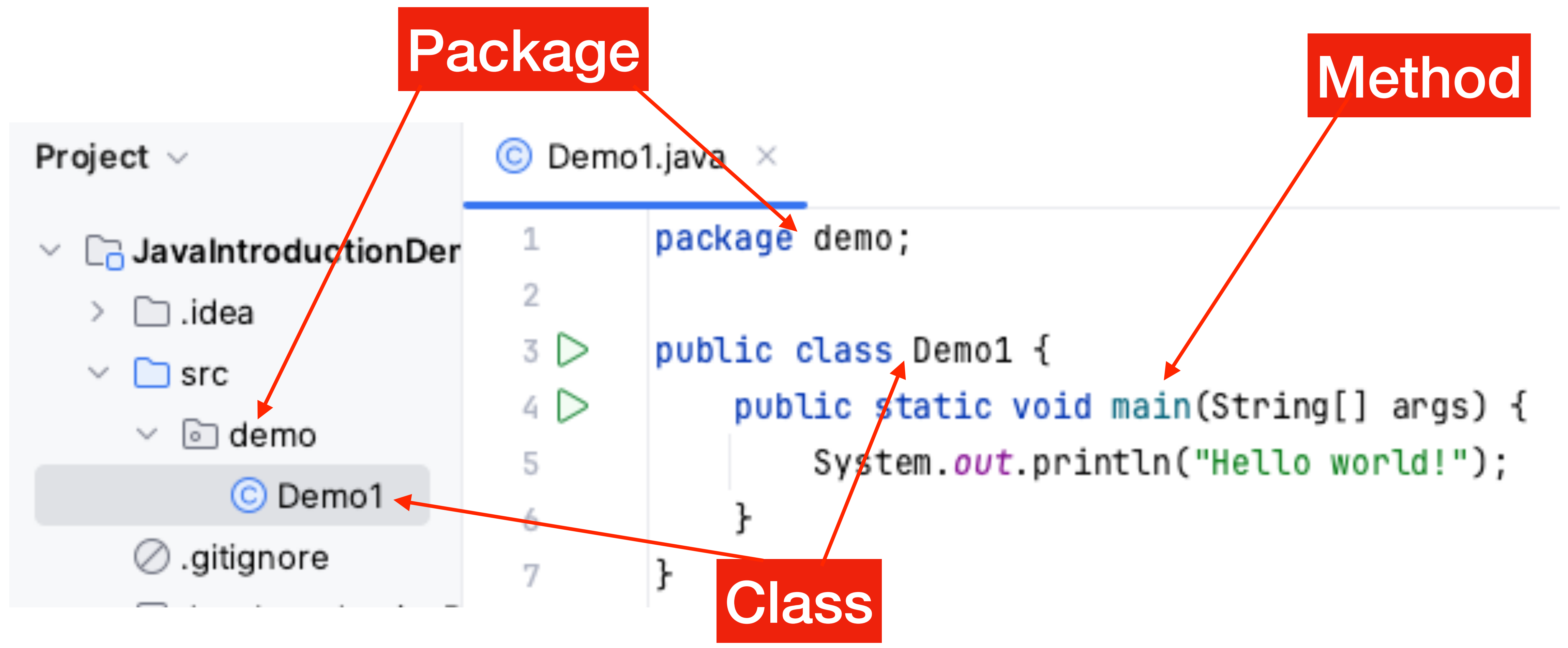
Packages, Classes and Methods

ACTLEA
Active Learning

Learning objectives

- Packages
- Classes
- Methods
- Access modifiers
- Varargs
- Method overloading
- Recursion

Basic structure of a Java program



```java
package demo;

public class Demo1 {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

- Methods - functions containing Java code that can be executed
- Classes - fundamental building blocks, contains the methods
- Packages - contains classes in a hierarchical structure

Packages

- Packages are used to organize and manage classes and prevent naming conflicts
- Related classes can be grouped within a package
- A package contains classes almost like a folder on a hard drive contain files
- Packages matter when it comes to Access modifiers

Classes

- A Java program typically consists of one or more classes
- Classes can contain methods and variables (behavior and data)
- Classes can also be used as blueprints for creating objects (OOP)

Methods

- Methods are like functions within classes, defines the behavior of the program
- Methods have a name, a return type and possibly input arguments
- Methods can be static (belong to the class) or non-static (belong to the object)
- Methods also have an access modifier (private, default, protected, public)

ACTLEA Active Learning

Access modifiers

Access modifier	Within Class	Within Package	Outside Package and in Subclass	Outside Package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Access modifiers

- Access modifiers apply to both methods and variables in the class
- (Local variables inside methods do not have access modifiers)

Demo 1 - Programming structure

- Packages
- Classes
- Methods
- Return types and Access modifiers

Variable arguments (varargs)

- Method input arguments that accept an arbitrary number of values
- The values will automatically be received as an array
- A method can only have one varargs argument
- The varargs must be last if the method has other input arguments

Demo 2 - varargs

- Defining a varargs argument
- Using a varargs argument

Exercise 1 - Revert an array

- Reuse exercise 1 from Arrays - Revert an array
- Refactor the code and put the code that reverts the array in a method
- Call the method from the main method, get the reverted array that is returned, and print it to the console

Exercise 2 - Revert numbers

- Reuse exercise 1- Revert an array, but this time do not use an original array, instead just send a comma separated list of numbers to a method that returns the reverted numbers in an array
- Use varargs in the input argument of the method, the return type should still be an int array
- You don't need to print the original numbers, just print the reverted array that is returned from the method

Overloaded methods

- Multiple methods with the same name but different parameters
- Input arguments can differ in number, type, or both
- The name of a method does not need to be unique
- The signature of a method needs to be unique (name and parameters)

Demo 3 - Overloaded methods

- Defining a varargs argument
- Using a varargs argument

Recursion

- Recursion is when a method calls itself
- Could be used for repetition instead of loops
- Has a recursive condition that calls itself
- Usually has a condition that stops the recursion and prevents an infinite loop

Demo 3 - Recursion

- Recursion example

Exercise 3 - Lowest, highest, sum, average

- Reuse exercise 2 from Arrays - Lowest, highest, sum, average
- Refactor the code and use four separate methods from the main method to calculate lowest value, highest value, sum and average

Exercise 4 - FizzBuzz

- Reuse exercise 4 from Repetition - FizzBuzz
- Still have the loop in the main method, but put the calculation of Fizz/Buzz/FizzBuzz/number in a separate method and call it from the loop
- The method should take an int as an input argument and return a String that can be printed to the console from the loop

Learning objectives

- Packages
- Classes
- Methods
- Access modifiers
- Varargs
- Method overloading
- Recursion

ACTLEA
Active Learning