Welcome to the

Java Course

Module 3 – Day 01

Content of the course

- **Object-Oriented Programming concepts**
- **Classes and objects**
- Inheritance and polymorphism
- Encapsulation and accessibility
- Exceptions

Air Flight Company Project

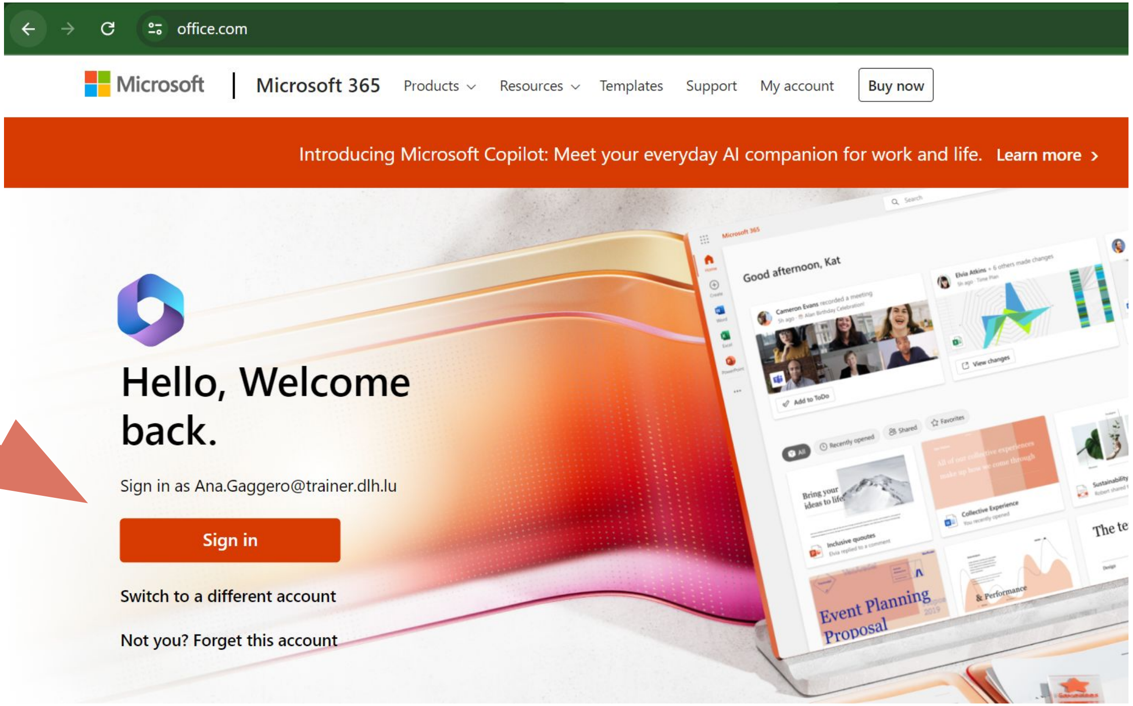
- **Manages flight** details and statuses.
- Manage different **types** of aircraft, commercial planes and cargo planes
- Flight **status** to track flight conditions.
- **Employee** Hierarchy representing staff.

Log into the local PC and to Microsoft Teams

Open Google Chrome and type: office.com

Step 1: Click on “Sign in”

You should have received your password by email.



Introducing Microsoft Copilot: Meet your everyday AI companion for work and life.

Hello, Welcome back.

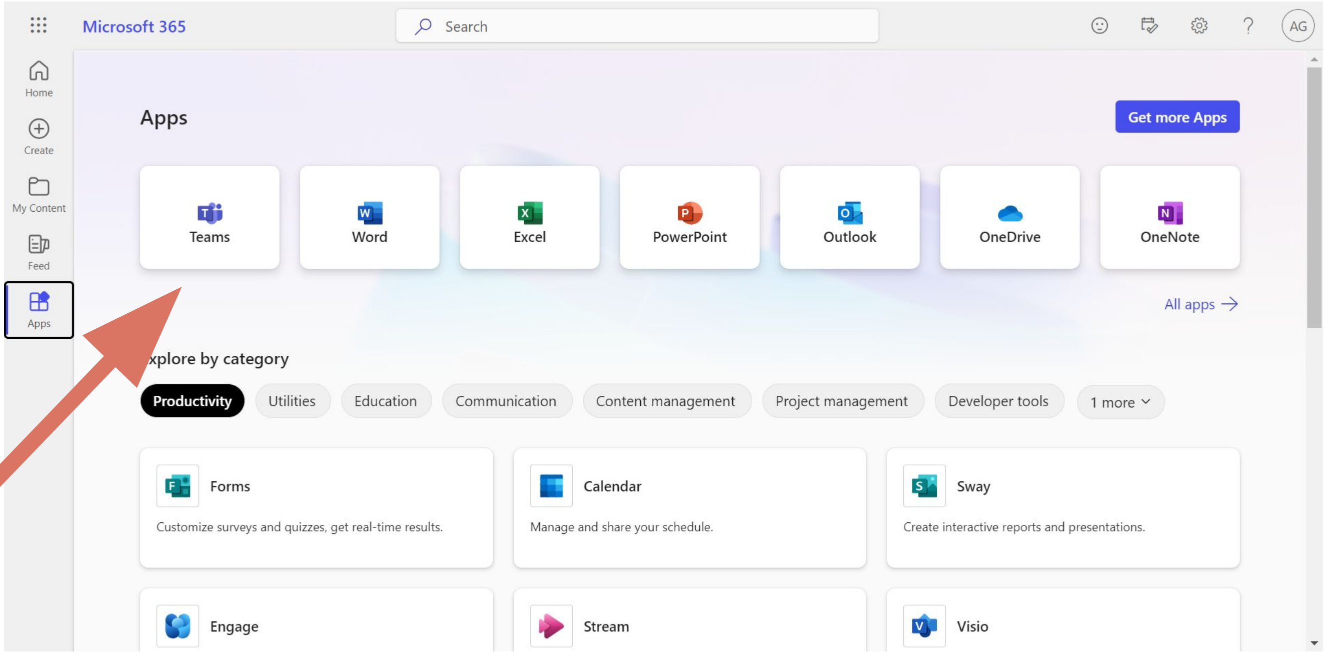
Sign in as Ana.Gaggero@trainer.dlh.lu

Log into the local PC and to Microsoft Teams

Step 2: Click on “Teams”

1

2



Customize surveys and quizzes, get real-time results.

Manage and share your schedule.

Create interactive reports and presentations.

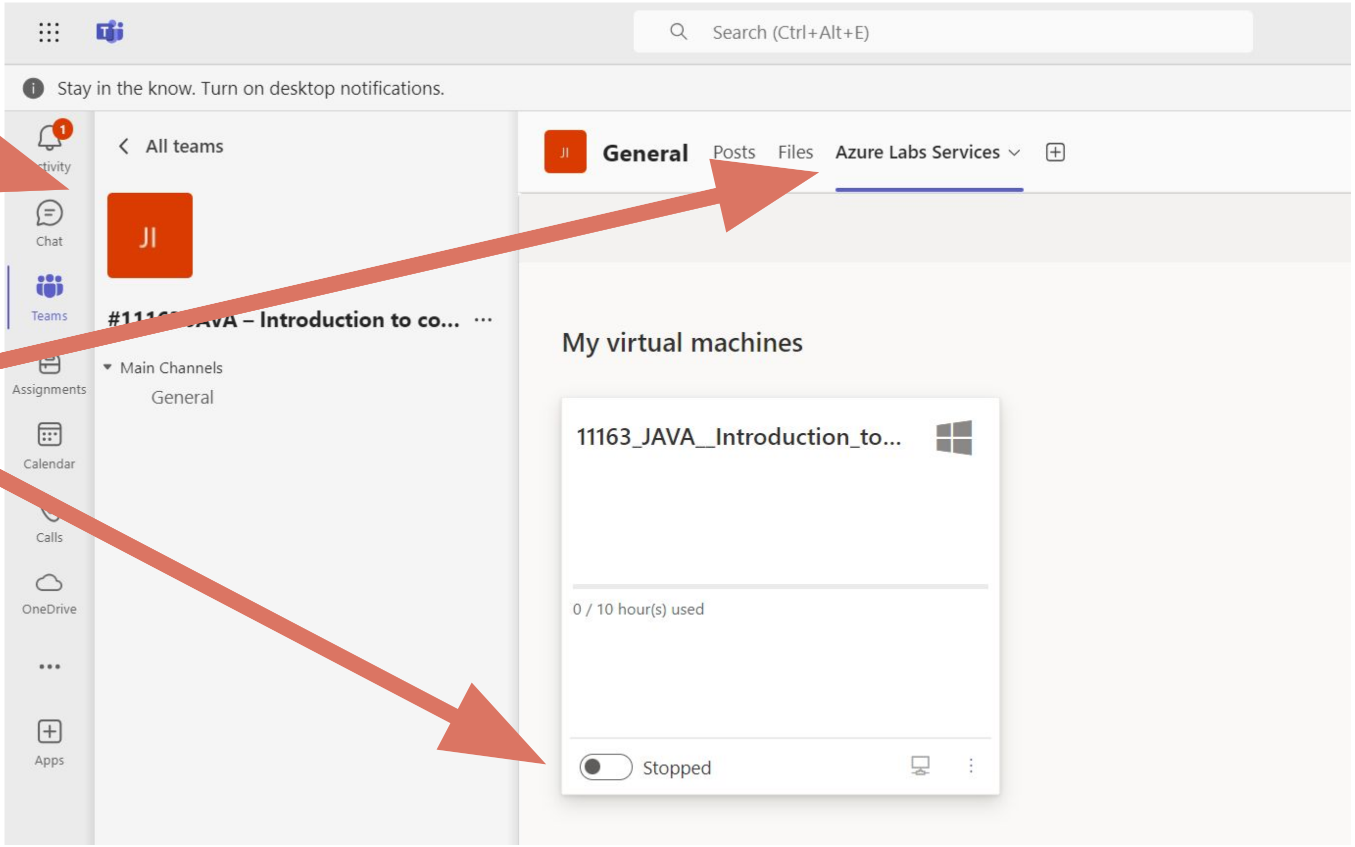
Log into the virtual machine

Step 1

Step 2

Step 3

The virtual machine will be already on. If not, turn it on, it can take up to 5 minutes to start.



Log into the virtual machine

Username: student
Password: StudentDLH2024

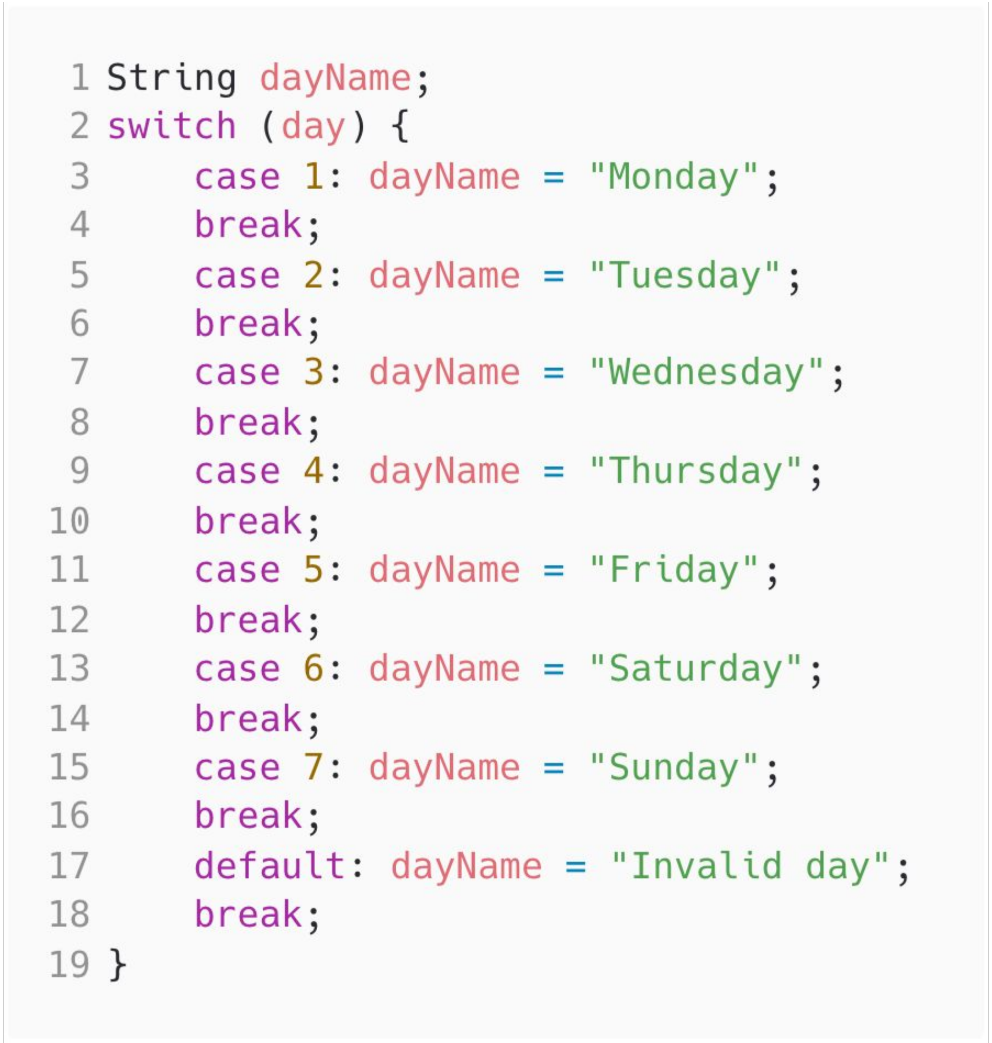
Recap of module 1 & 2

- Conditionals
- Switch
- Loop while
- Loop for
- Functions and procedures
- Arrays
- List
- Map

Nested Conditionals

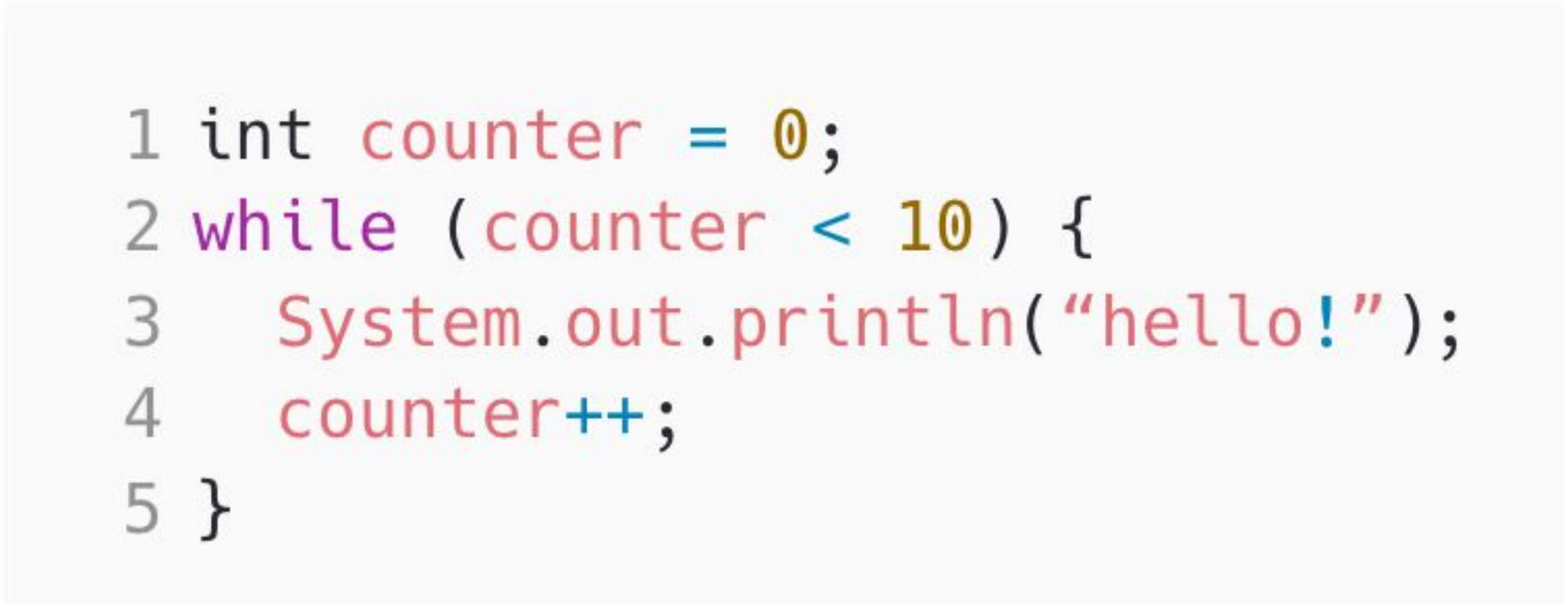
```java
// Check age group
if (age < 18) {
  System.out.println("You are a minor.");
} else {
  if (age < 65) {
    System.out.println("You are an adult.");
  } else {
    System.out.println("You are a senior.");
  }
}
```

Switch



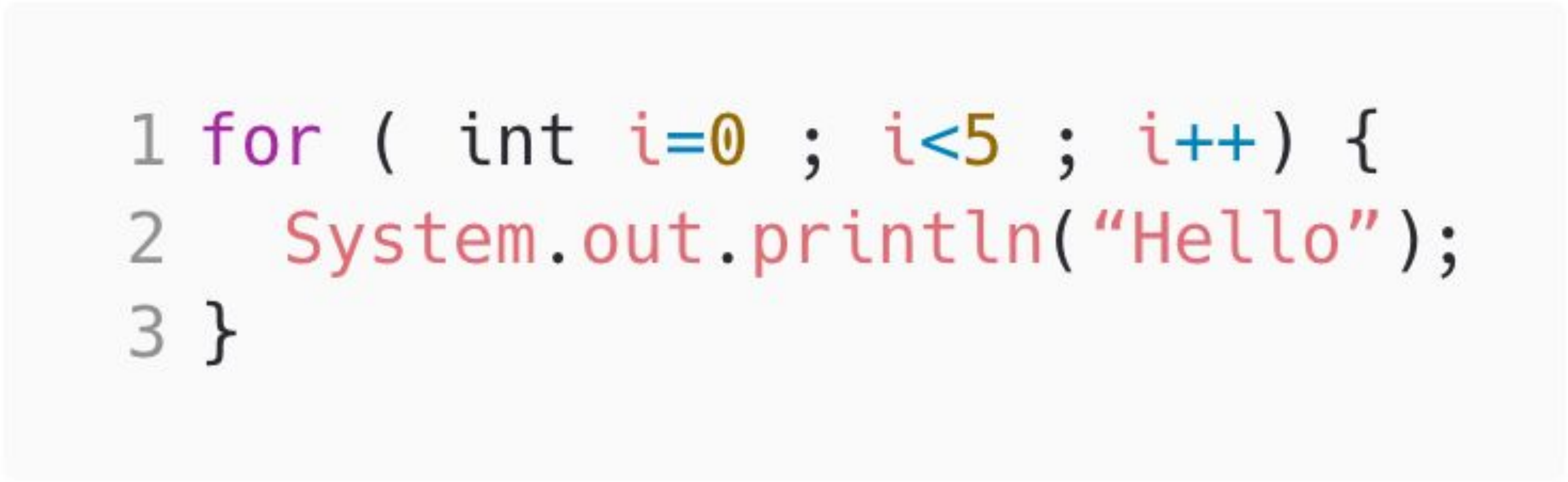
```
String dayName;
switch (day) {
     case 1: dayName = "Monday";
     break;
     case 2: dayName = "Tuesday";
     break;
     case 3: dayName = "Wednesday";
     break;
     case 4: dayName = "Thursday";
     break;
     case 5: dayName = "Friday";
     break;
     case 6: dayName = "Saturday";
     break;
     case 7: dayName = "Sunday";
     break;
     default: dayName = "Invalid day";
     break;
}
```

Loop while



```
int counter = 0;
while (counter < 10) {
  System.out.println(“hello!”);
  counter++;
}
```

For Loop

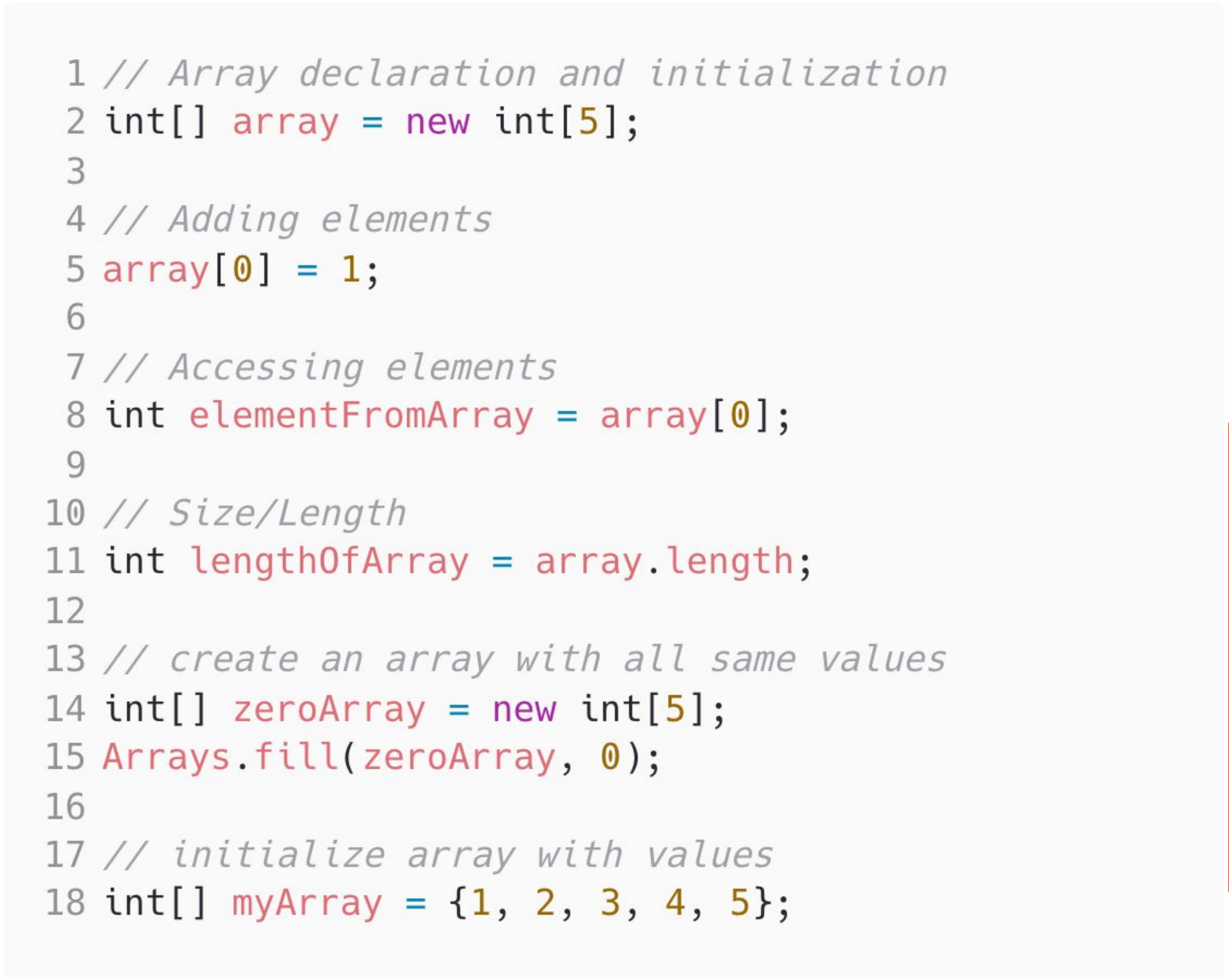


```
for ( int i=0 ; i<5 ; i++) {
  System.out.println(“Hello”);
}
```

Functions & Procedures

```
public class Main {

    public static void main(String[] args) {
            // block of code
    }

    public static int addNumbers(int num1, int num2) {


            return num1 + num2;

    }

}
```

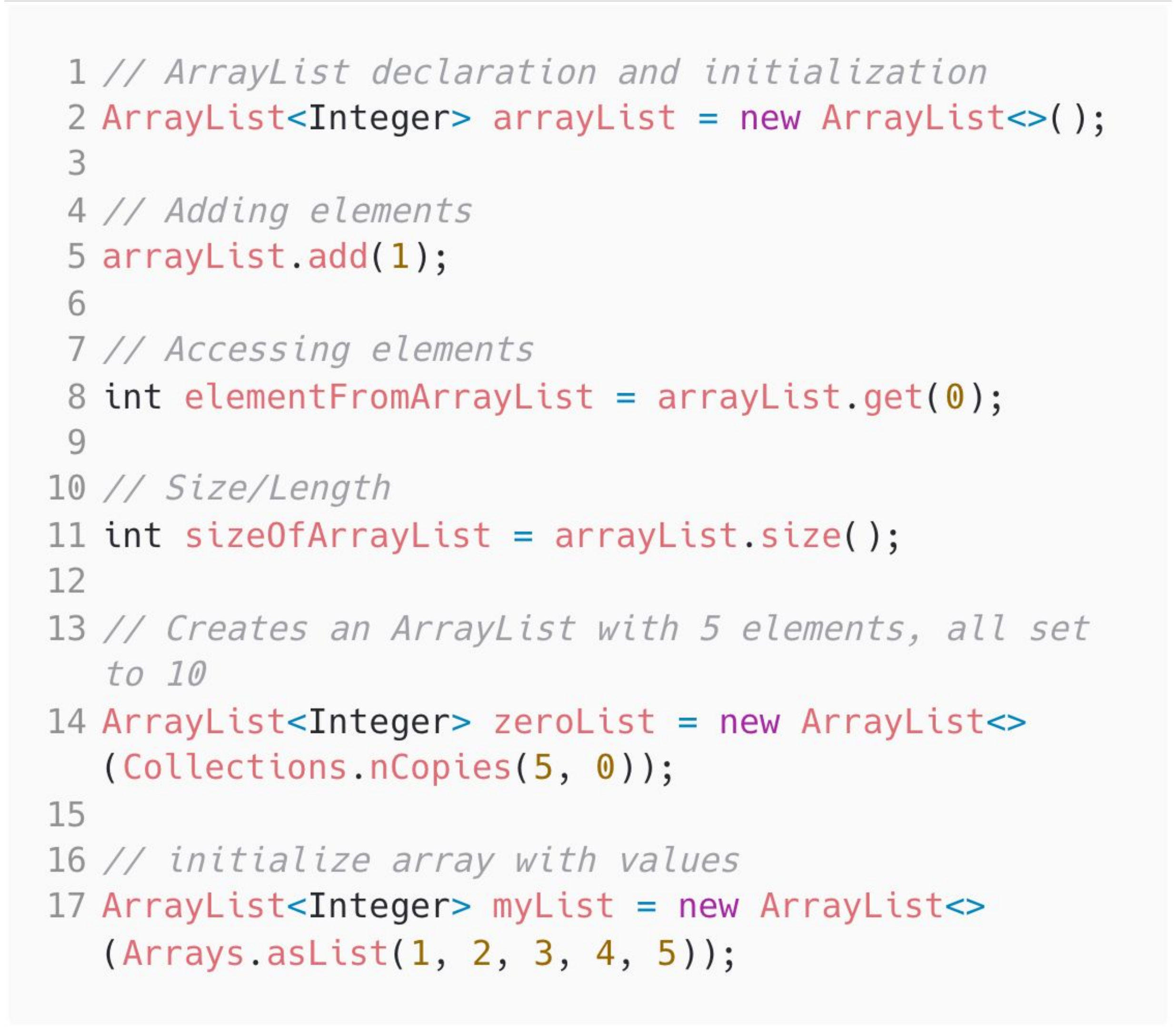
Arrays



```java
// Array declaration and initialization
int[] array = new int[5];

// Adding elements
array[0] = 1;

// Accessing elements
int elementFromArray = array[0];

// Size/Length
int lengthOfArray = array.length;

// create an array with all same values
int[] zeroArray = new int[5];
Arrays.fill(zeroArray, 0);

// initialize array with values
int[] myArray = {1, 2, 3, 4, 5};
```

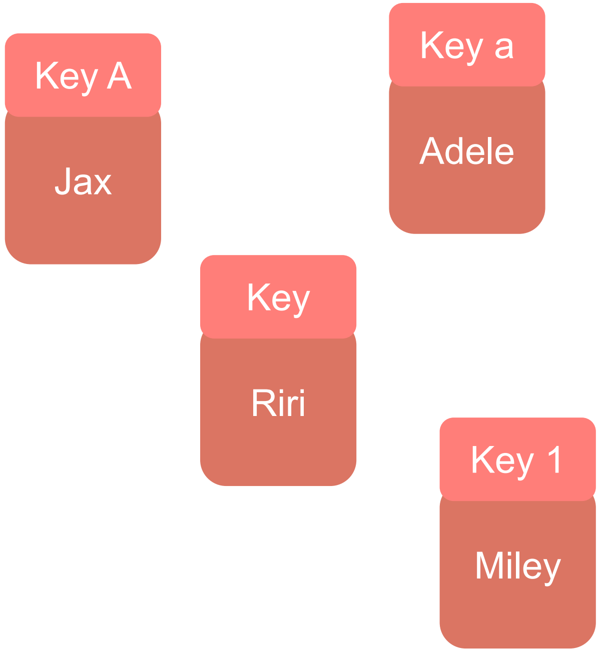
Lists

- ArrayList<>



```java
// ArrayList declaration and initialization
ArrayList<Integer> arrayList = new ArrayList<>();

// Adding elements
arrayList.add(1);

// Accessing elements
int elementFromArrayList = arrayList.get(0);

// Size/Length
int sizeOfArrayList = arrayList.size();

// Creates an ArrayList with 5 elements, all set to 10
ArrayList<Integer> zeroList = new ArrayList<>(Collections.nCopies(5, 0));

// initialize array with values
ArrayList<Integer> myList = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));
```

Map



Example

```
// Create a map to store student IDs and their
names
Map<Integer, String> students = new
HashMap<>();
// Add students to the map
students.put(1035, "Alice");
students.put(1037, "Bob");
students.put(1038, "Charlie");
//Print the students
System.out.println("Students: " + students);
// Remove a student by their ID and print the
students again
students.remove(1037);
System.out.println("Students: " + students);
```

Output

```
Students: {1035=Alice, 1037=Bob, 1038=Charlie}
Students: {1035=Alice, 1038=Charlie}
```

OOP

Object Oriented Programming is a different way to concept the code and the world around us.

OOP

Everything can be an object, even an abstract concept. The point is considering anything as an object.

OOP

The CLASS is a generic object that represents a category of things.

For EXAMPLE :

DOG

Not a specific dog, but the concept of a generic dog.

OOP

```
public class ClassName
{
}
```

OOP

```java
public class Dog {
}
```

Class

- Attributes
- Methods

Dog ATTRIBUTES:

```
public class Dog {

  public int size;
  public String colour;
  public int age;

}
```

Now **YOUR TURN !**

Let’s do exercise 1

OOP

Functions defined into a class are called METHODS. They are the actions that the object can execute.

Dog METHOD:

```
public void bark() {
  System.out.println("woof!");
}
```

OOP

The method aimed to the **attributes initialization** is called CONSTRUCTOR. Using it, you can define the attribute values.

Dog CONSTRUCTOR:

```
public Dog(){
  size = 0;
  colour = "";
  age = 0;
}
```

Dog CONSTRUCTOR:

```
public Dog(int size, String colour, int
age){
  this.size = size;
  this.colour = colour;
  this.age = age;
}
```

OOP

An INSTANCE is a specific object of a class, with specific property values. It's not a generic object, but an object with a name and an identity.

OOP

```
Dog lola = new Dog();

Dog nala = new Dog(50, "Beige", 2);
```

Calling Method

$$\underbrace{\texttt{nala}}_{\text{Instance}}\texttt{.}\underbrace{\texttt{bark}}_{\text{Method}}\texttt{();}$$

Now YOUR TURN !

Let’s do exercise 2

Dog ATTRIBUTES:

```
public class Dog {

  private int size;
  private String colour;
  private int age;

}
```

Getter

```
public int getSize() {
  return size;
}

public String getColour() {
return colour;
}

public int getAge() {
return age;
}
```

Setter

```
public void setSize(int size) {
  this.size = size;
}

public void setColour(String colour) {
  this.colour = colour;
}

public void setAge(int age) {
  this.age = age;
}
```

Now YOUR TURN !

Let’s do exercise 3

Air Flight Company Project - Step 1

- Create a “Flight” class with properties to store the following information:
    - Flight number
    - Destination
    - Capacity
    - Amount of booked seats

- All properties should be private and the class should have get and set methods for them.

Air Flight Company Project - Step 1

- Whenever a new Flight gets created, it should have no booked seats.

- The “Flight” class should provide a method to allow booking a seat. This method will return true if it was possible to book a seat and will increment the amount of booked seats for the flight. It will return false if it was not possible to book a seat because the flight is already full.

Air Flight Company Project - Step 1

```
>>> New Flight <<<
Enter flight number: 3527
Enter destination: Madrid
Enter flight capacity: 180
Flight created.

Would you like to (a) book a seat or (b) see the amount of available
seats? a
Seat booked!
Would you like to (a) book a seat or (b) see the amount of available
seats? a
Seat booked!
Would you like to (a) book a seat or (b) see the amount of available
seats? b
Available seats on flight 3527 to Madrid: 178
```