Lab1: Classes & Objects
(3h)

Instructions:

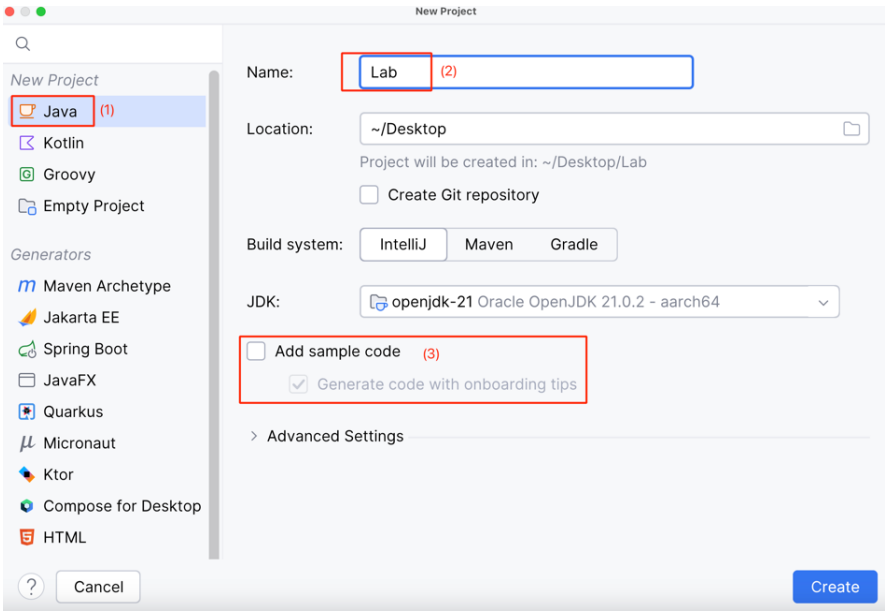
⇒ This work is INDIVIDUAL
⇒ The IDE to use is IntelliJ
⇒ It is strongly recommended to open the CM support during the whole Lab1 session

Learning Outcome:

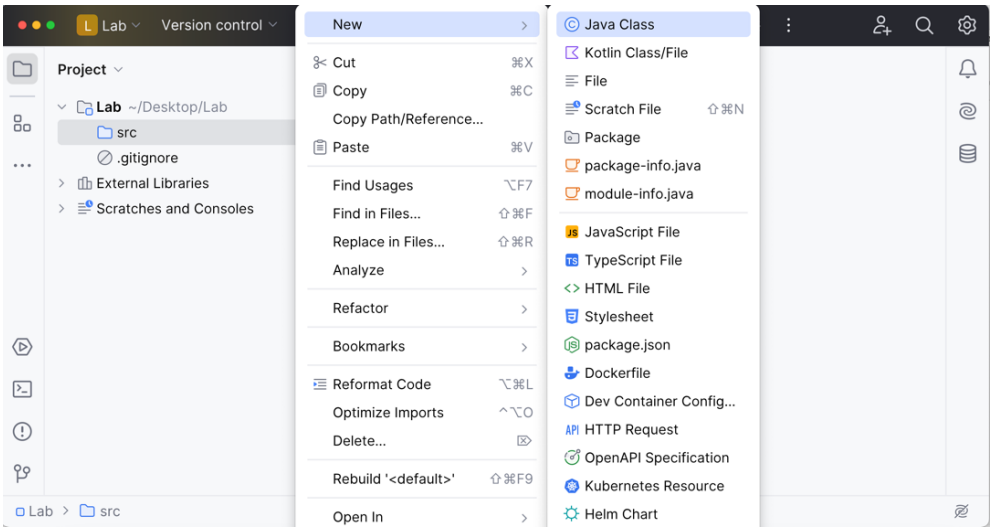
At the end of this Lab, each student should be able to:
- Create a Java class
- Define the attributes of a Java class
- Define the constructors of a Java class
- Define class methods
- Instantiate an object
- Modify the attributes of a class
- Display an Object

Creating a New IntelliJ Project

1. Open IntelliJ IDEA.
2. Click on New Project.
3. Select **Java**(1) as the project type.
4. Give the **projet a name**(2) and choose **a location**.
5. Make sure the JDK is selected (if not, click Add JDK and choose an installed version).
6. Uncheck the option ***Add sample code***(3).
7. Click on **Create**.

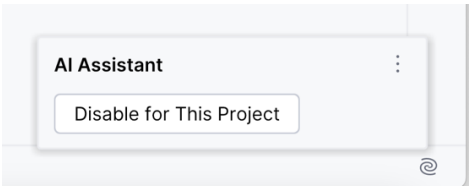


8. In the **Project Explorer** (top left), right-click on the **src** folder.
9. Select **New > Java Class**.
10. Enter the **class name** (e.g., Main).
11. Press Enter.



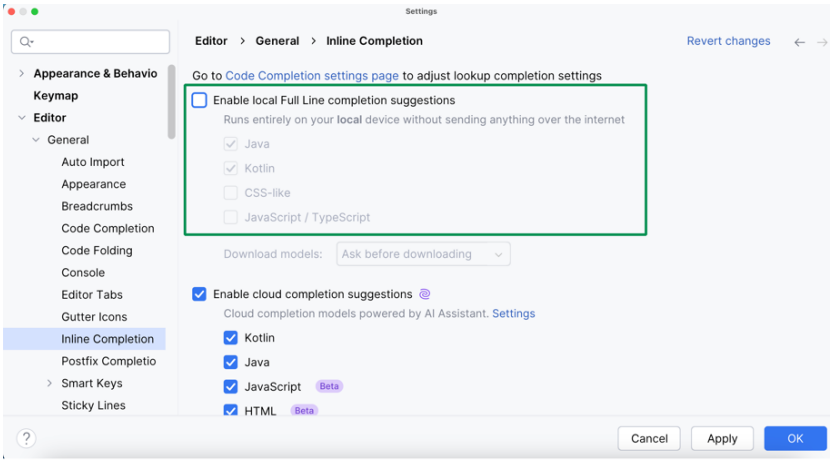
To effectively learn the Java programming language, it is **essential** to **disable artificial intelligence** and the **automatic code completion feature**. This will help you develop your logic and coding autonomy.

Step 1: At the bottom right, click on the **AI icon** and disable the **AI Assistant** option.



Step 2: **Disable Automatic Completion**

- Go to **File** (or IntelliJ IDEA on Mac) > **Settings** > **Editor** > **General** > **Inline Completion.**
- **Uncheck** the option ***Enable Local Full Line Completion Suggestions***.
- Click **Apply**, then **OK**.



Organization of the code

Create **a new file for each of the** requested **classes.**

Beginner level

Exercise 1

1. Create a **Point** class and a **Main** class.
2. Code the exercise 1 of the programming part of Tuto1.
3. Add to the **Point** class a **display()** method allowing to display a point under the form (x, y)
4. Add to the **Point** class a static method **distanceStat** allowing to calculate the Euclidean distance between two points given in parameters.

   Reminder:

   `P1(x1, y1) , P2(x2, y2)` ➔ `distance (P1, P2) = (x1-x2)² + (y1-y2)²`
5. Add to the Point class a **distance()** method that calculates the Euclidean distance between the point of the current instance and a point given in parameters.
6. Test in the Main class the two methods **distanceStat** and **distance()**
7. Add the file *Rectangle.java* to your project.
8. Code the exercise 2 of the Programming part of Tuto1

Exercise 2

1. Create a **Date** class and a **Main** class.
2. Add to the **Date** class three attributes of type integer: day, month, year
3. Add to the **Date** class a constructor allowing to initialize the attributes of a given date
4. Add to the **Date** class a **display()** method allowing to display the date in the form ***day/month/year***
5. Add to the Date class a static method **compareDatesStat** which takes as parameters two variables **d1** and **d2** of type Date and returns 1 if **d1** is later than **d2,** -1 if **d1** is earlier than **d2** and 0 otherwise (equality).
6. Add to the Date class a method **compareDates** which takes as parameters a variable **d** of type Date and returns 1 if the date of the current instance is later than **d**, -1 if the date of the current instance is earlier than **d** and 0 otherwise (equality).
7. In the Main class, ask the user to enter three integers **j, m** and **a**
8. Create a Date instance
9. Display the Date object
10. Create a second object of type Date, representing the current date.
11. Compare it to the first object using the two methods **compareDatesStat** and **compareDates**, display a message accordingly.

Intermediate level

Exercise 1

A vehicle is represented by a string (its ***brand***) and three integers: the ***fiscal power***, the ***maximum speed*** and the ***current speed***.

All these data are representative of a particular vehicle, in other words, each vehicle object will have its own copy of its data in memory: this is called an instance attribute.

The instantiation operation that allows to create an object from a class consists precisely in providing particular values for each of the instance attributes.

1. Create the **Vehicule** class in the *Vehicule*.java file
2. The ***fiscal power*** of a vehicle, once initialized, cannot be modified. It is a constant. What is the keyword in Java for a constant ? add it to the class.
3. Define the other attributes of the class.
4. Add a default constructor to the **Vehicle** class**.** An error appears. Why does it appear? Fix this error.
5. Add a constructor with three parameters.
6. A class attribute ***number*** allows to count the number of vehicles present at a given time in the class. This is a typical example of an attribute shared by all the objects of the same class. It is not necessary that each object has its own copy of this attribute, it is better that they share a unique copy located at the level of the class.

   In which methods is the ***number*** class attribute updated? Specify what is done during this update.
7. We want to add the two methods ***start()*** and ***stop()*** to the class. Which attribute do you propose to add to the class that will be modified by these two methods? Create the methods and the attribute.
8. Add to the **Vehicle** class a **display()** method allowing to display all the information of a Vehicle object in the form :

```
Brand:
Fiscal Power:
Maximum speed:
Current speed :
```

9. In the **Main** class, declare a variable " **myVehicle** " of type **Vehicle**
10. Ask the user to enter its brand and maximum speed and modify its attribute information.
11. Turn off the "**myVehicle**" vehicle
12. Start the vehicle "**myVehicle**
13. Ask the user to enter the information of a second vehicle "**yourVehicle**" and create it.
14. Stop the vehicle "**yourVehicle**" if it is started, and start it if it is stopped.
15. Display the information of the two previous vehicles.
16. Display the value of the variable ***number***

Exercise 2

In this exercise, we will create **BankAccount** and **Customer** classes to simulate some real bank transactions.

BankAccount class

We define the **BankAccount** class which is composed of:

Attributes: ***numAccount***: is the unique identifier of the account

balance: represents the available balance on the account

interest: is the interest rate proposed by the bank with a default value set to 0.05.

Methods: ***changeInterestRate()***: associates an interest rate to an account number.

addInteret(): calculates and adds the accumulated interest to the account balance.

displayBalance(): prints the available balance of the account.

Write the default constructor and the constructor allowing to initialize all the attributes of the **BankAccount** class.

Customer class

We define the **Customer** class which is composed of :

Attributes: ***id*** : is the unique identifier of the client

lastname: the client's last name

firstname: the client's first name.

account: each client is entitled to only one bank account.

Methods: ***deposit (amount)***: deposits the amount in the bank account

withdraw (amount): withdraws the amount from the bank account.

Add all the indicated methods, modifying the BankAccount class if necessary.

Add two different constructors to the **Customer** class.

Add to the class a method that displays all the information of a customer.

Class **Application**

Create a main class called **Application** containing the *main* method and add the following:

1. Take the information of 3 clients of the user and create them. Use for that an array of size 3 of type **Customer**.

2. The second customer has deposited a sum of 5,000 euros in his/her account, the bank then increases its interest rate to 0.1.
3. Display the information of the second client.
4. Display the available balance on each of the clients' accounts using a for loop.