# Tutorial 7: Classes & Objects

Aim:

- Get familiar with classes and objects.
- Consolidate learning from week 7.
- Get feedback.

Note: Use the material from the lecture if you need. If you find any problem or have a question, ask your tutor. If you do not have enough time during the session, it is recommended that you finish the exercises at home. Challenges are optional, although recommended.

## Section 01: Main Questions

1. Create a new class called **Student which is empty.**

```
public class Student {

}
```

2. Add the following private attributes (instance variables) to the class: student name (String) and student ID (String). You should have something like this:

```
public class Student{
     private String name;
     private String ID;
}
```

3. Add a constructor to your class that creates a new student given a name and an ID:

```
public Student(String name, String ID){
     this.name = name;
     this.ID = ID;
}
```

4. Add a public method in your Student class to get the name of the student (getter):

```
public String getName(){
     return this.name;
}
```

5. Add a public method in your Student class to change their name (setter):

```
public void setName(String name){
     this.name = name;
}
```

Make sure you understand the use of this in the previous code.

6. In your main, create a new student, print their name, change their name and print it again:

```
public static void main(String[] args) {
      Student s1 = new Student("Alex", "w12345");
      System.out.println(s1.getName());
      s1.setName("Bob");
      System.out.println(s1.getName());
}
```

7. Add the getter and setter for ID in the class Student.

**Q2: Define another class.**

Add a new class called **Module** to your project. The class Module should have a module code (String), a mark for the in-class test (double) and a mark for coursework (double). Add a constructor to your Module class with the 3 attributes as parameters.

**Q3: Add methods to a class.**

- In your new class **Module**, add a **private** method called `getFinalMark` that calculates and returns the final mark of the module. The in-class test is worth 50% and the coursework also 50%.
- Add a **public** method to your class Module called `pass()` that uses the method `getFinalMark` to get the final mark and checks if the student will pass the module or not. If the final mark is larger or equal to 40, the student will pass the module and the method should return `true`, otherwise the method should return `false`.
- Add the getter method to get the code of the Module.

**Q4: Update Student.**

Update your class Student by adding an array of 6 Modules as an attribute:

```
private Module[] modules;
```

You will also need to update the constructor of Student to initialise the list of modules:

```
public Student(String name, String ID){
      this.name = name;
      this.ID = ID;
      modules = new Module[6];
}
```

2.- Add a public method to add a Module to the Student. Remember that your array of type Module will contain 6 pointers pointing to `null` at initialisation time. When adding a Module, you must check if there is space in the array and find a position in the array to insert the Module instance.

3.- Add another public method to Student called `showPass()` that for each Module in the `modules` array uses the `pass()` method in Module to print a message stating if the student will pass the module or not.

**Q5. Create new Students and Modules.**

In your `main`, create multiple students with different names and IDs. Create multiple modules for each student and add the modules to the students. For each student, call the method `showCall` to print a summary showing which modules the students will pass and fail.

## Section 02: Challenging Questions

**Q6: Creating Bank Account class**

Create a class called **Account** where you have a **Holder name, Account number, Account type and current balance**.

a. Implement a **constructor** to initialize Account objects. Use appropriate data types for attributes as you see fit.

b. Implement the following methods to handle account behaviour.

- To deposit an amount. ( `void depostMoney`)
- To withdraw an amount. (`void withdrawMoney`)
- To display the name, account type and account number. (`void displayAccountDetails`)
- To check the current balance. `(double checkBalance)`

8. Creates a driver class called **AccountTest** with the main method.

   a. Prompt the user for Holder name, Account number, Account type and initial balance to create an Account object from the Account class created in question 1.

```
Enter Account Holder's Name:
John Doe
Enter Account Number:
12345678
Enter Initial Balance:
999.99
Enter Account Type:
Savings
Creating John Doe's Account
```

b. Prompt the following options continually for the user until the user wishes to exit from the system. Use a switch-case construct. refer to the following sample outputs.

```
Enter Account Holder's Name:
John Doe
Enter Account Number:
12345678
Enter Initial Balance:
500.0
Enter Account Type:
Current

Account Created


To Withdraw Money          :Press 1
To Deposit Money           :Press 2
To Check Balance           :Press 3
To View Account Details    :Press 4
To Exit                    :Press 5
```

```
Enter the Amount you want to withdraw:

300.0

To View Account Details    :Press 4
To Exit                    :Press 5
Current Account Balance is : 200.0

To Withdraw Money          :Press 1
To Deposit Money           :Press 2
To Check Balance           :Press 3
To View Account Details    :Press 4
To Exit                    :Press 5
```

c. Implement all the validations for transactions.

**Q7: GP surgery.**

A GP surgery has asked you to create a Java program to keep information regarding patients and appointments. Create at least 2 classes, with their methods, getters and setters. Which classes would you use? Which attributes will they have? Design and implement the application using object-oriented programming.

## Section 3:HackerRank Interview questions.

You can solve the following tasks in HackerRank:

1. Java Inheritance l (inheritance was not explained during the lecture but they provide an example)
2. Recursive digit sum (not object-oriented programming)