Challenge 3: Implement a Father Class

In this challenge, we'll implement a base class father and derived classes, son and daughter.

WE'LL COVER THE FOLLOWING

- Problem Statement
  - Input
  - Sample Input
  - Expected Output
- Coding Exercise
  - Solution Review

Problem Statement

Implement a code which have:

- A **parent** *class* named `Father`.
  - Inside it *define*:
    - `eye_color`
    - `hair_color`
    - `void Father_traits()` function:
      - It prints the eye_color and hair_color of the called object
- Then, there are **two derived** *classes*
  - `Son` class
    - has a *private* member `name`
    - has a function named `Son_traits()` which prints traits of the Son
  - `Daughter` class
    - has a *private* member `name`

        - has a function named `Daughter_traits()` which prints traits of the Daughter
- The **derived classes** should
    - call the method of the `Father` class which prints the `eye_color` and the `hair_color` and for `Son` and `Daughter` classes prints the *name* of a respective object.

Input

- In `Son` class, `eye_color` is set to **Brown** and the `hair_color` is set to **Black** and `name` is set to **Ralph** in parametrized constructor of `Son` object
- In `Daughter` class, `eye_color` is set to **Green** and the `hair_color` is set to **Golden** and `name` is set to **Rapunzel** in parametrized constructor of `Daughter` object
- Now, print `Son_traits` and `Daughter_traits` from their respective objects

Here's a sample result which you should get.

Sample Input

```
Daughter obj("Rapunzel","Green","Golden");
obj.Daughter_traits();

Son Obj("Ralph","Brown","Black");
Obj.Son_traits();
```

Expected Output

```
Eye color: Green
Hair color: Golden
Rapunzel have long hairs!

Eye color: Brown
Hair color: Black
Ralph have beard!
```

Coding Exercise

Implement the code in the **problem** tab.

Good Luck!

Exercise | Solution

```
#include <iostream>
using namespace std;

// Write your classes here

int main() {
  // create classes objects here
  // call derived class member functions here
  return 0;
}
```

Show Hint

Solution Review

- We have implemented `Father` class which have **eye_color** and **hair_color** variables, and a function **Father_traits()** which prints *eye_color* and *hair_color* of animal
- Now implement `Daughter` and `Son` classes inherited publicly from `Father` class
- `Daughter` has private string **name** variable and a function **Daughter_traits()** which calls **Father_trais()** function and prints *name* of the Daughter
- `Son` has private string **name** variable and a function **Son_traits()** which calls **Father_trais()** function and prints *name* of the Son
- Create *Son* and *Daughter* object by calling parametrized constructors of the classes and print their traits by calling their respective functions

In the next chapter, we'll learn about a very important concept of OOP paradigm, `polymorphism`.