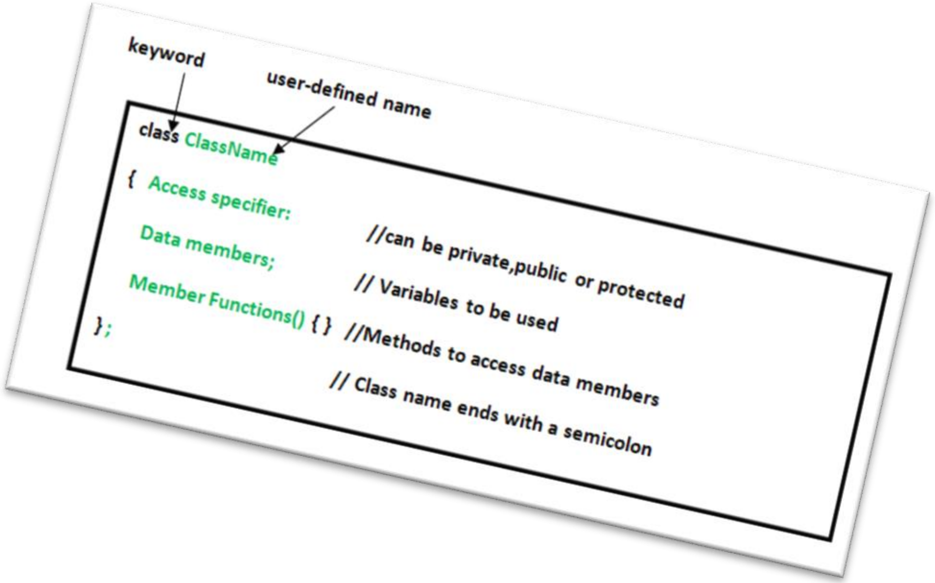
TODO BEFORE

Preparation before next session

⚠ *You will need to present your research during the next session*

LEARN

1. Read W3C course about classes in C++
2. Read this documentation also



```
class ClassName
{  Access specifier:          //can be private,public or protected
   Data members;              // Variables to be used
   Member Functions() { }     //Methods to access data members
};                            // Class name ends with a semicolon
```

RESEARCH

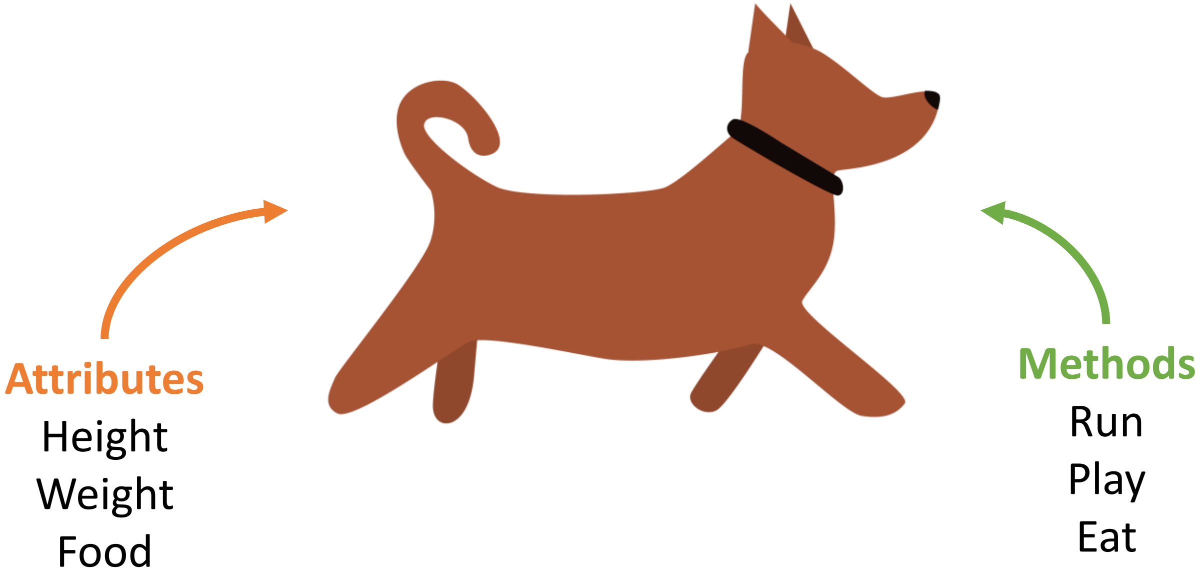
Make researches on the following points

Q1 – what is the difference between a **class** and an **object** ?

Q2 - What are the **benefits** of using classes ?

ALGORITHM ADVANCED

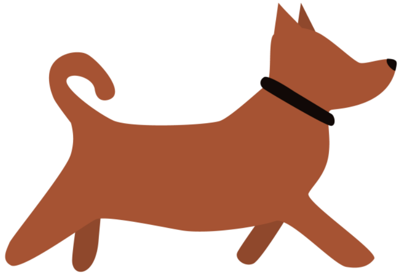
C1-S4 – Class & Objects



CADT
IDT

Session Objectives

- ✓ **Benefits of class** vs structures
- ✓ Class **attributes** & methods
- ✓ Class **instantiation** and constructors
- ✓ Class members **visibility**
- ✓ **this** as a reference to current object
- ✓ Pass objects by **value** vs **reference**
- 🔍 *(research activity)*
  Dynamic instantiation & destructors



PRESENTATION

🕒 10 MIN

Present your research activities

Discussion groups about your homework research

Q1 – what is the difference between a **class** and an **object** ?

Q2 - What are the **benefits** of using classes ?

EXPLAIN

A **BankAcount** *using a structure*

In a structure you define only the **structure attributes**

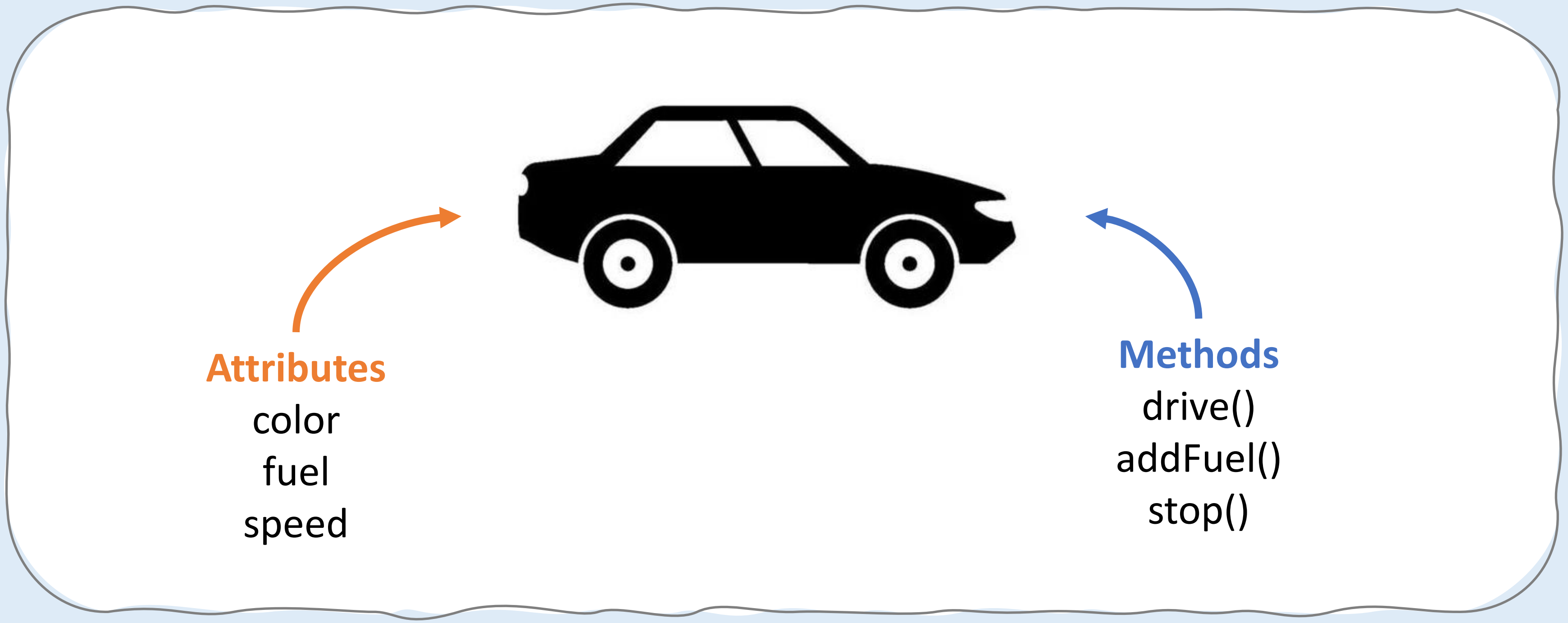
Functions manipulating data are defined outside the structure

- Here, we use a **structure** to define a BankAccount and store its balance
- But the structure **does not prevent users** to modify the balance incorrectly

```
struct BankAccount {
    double balance;
};

int main() {
    BankAccount account;
    account.balance = -50.0; // Ooops ! The balance is negative !

    std::cout << "Account Balance: " << account.balance << std::endl;
    return 0;
}
```

What about Object-Oriented ?

It would be nice if functions manipulating a structure were automatically associated with this structure



Attributes

color

fuel

speed

Methods

drive()

addFuel()

stop()

That the goal of classes ! to gather attributes and functions in a same place

EXPLAIN

BankAcount - *using classes*

In a class we can define members which can be attribute or methods (functions)

- Here, we use a **class** to define a BankAccount and store its **balance**
- The class contains not only the balance but also **a function** to manipulate this data properly

```
class BankAccount {

private:
    double balance;

public:
    BankAccount(double initialBalance) : balance(initialBalance) {}

    void withdraw(double amount) {
        if (amount > balance) {
            throw std::runtime_error("Insufficient funds for withdrawal.");
        }
        balance -= amount;
    }
};
```

A CLASS

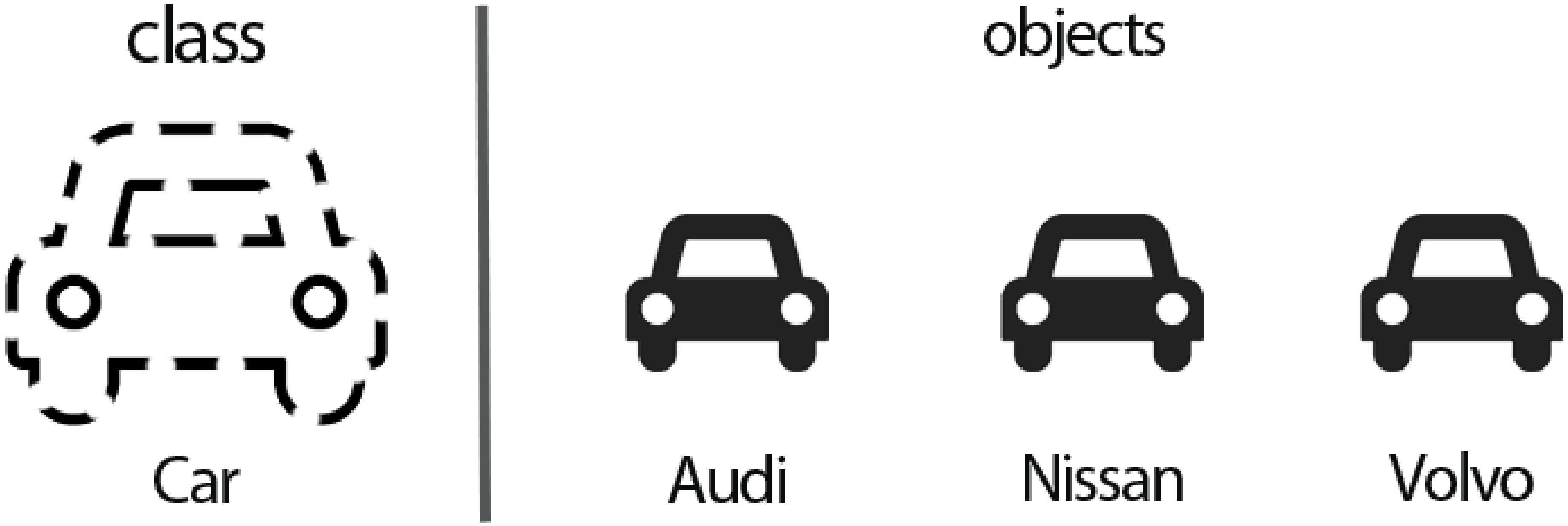
IS A

BLUEPRINT

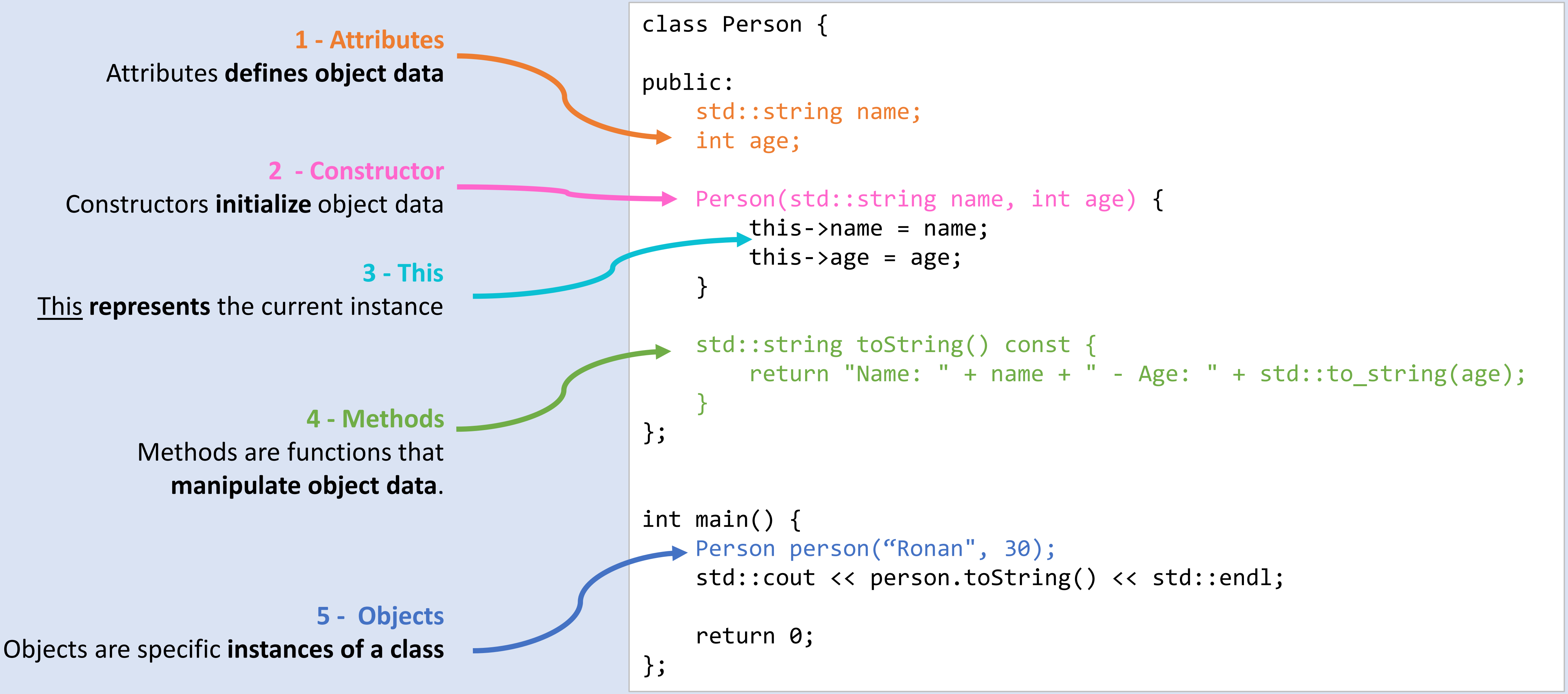
FOR

OBJECTS

Form 1 model (class) you can create many variation (objects)



Key Concepts



1 - Attributes

Attributes **defines object data**

2 - Constructor

Constructors **initialize** object data

3 - This

This **represents** the current instance

4 - Methods

Methods are functions that **manipulate object data**.

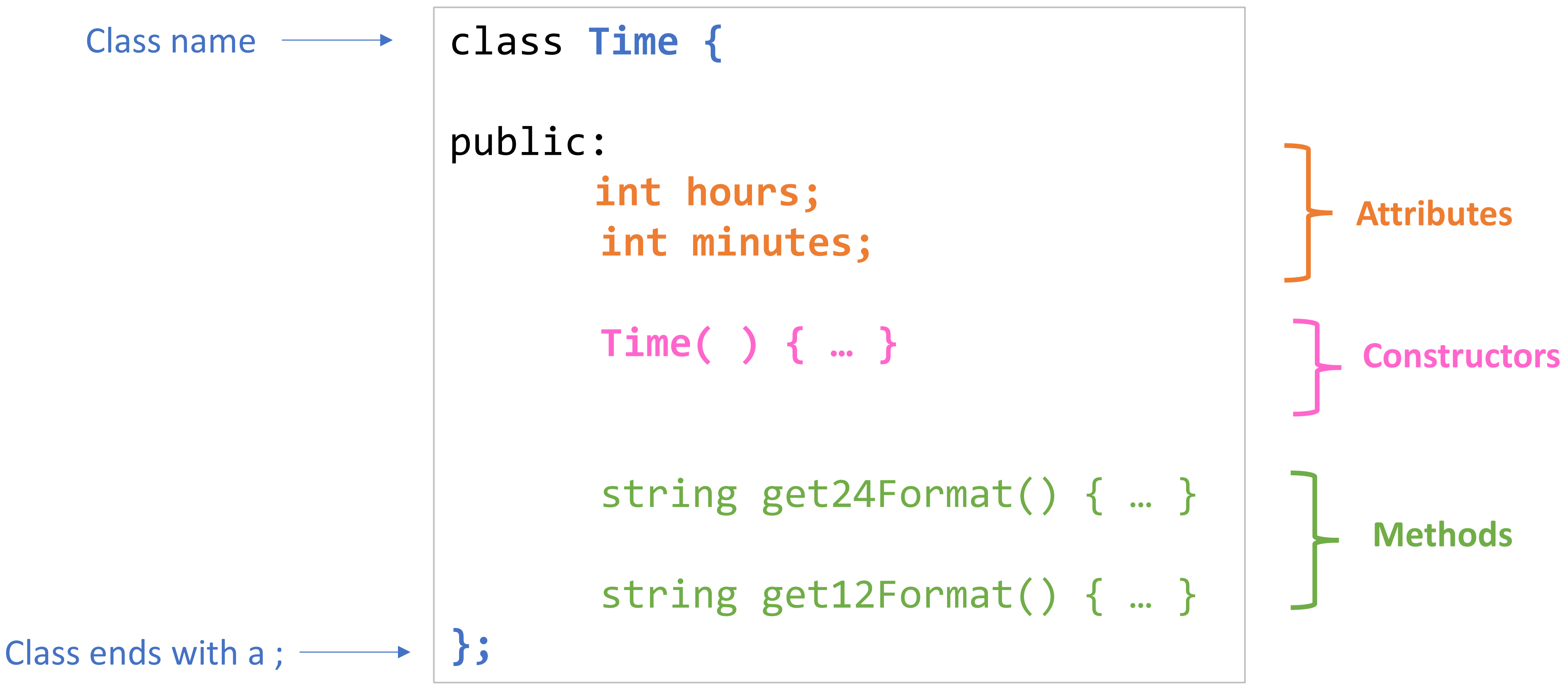
5 - Objects

Objects are specific **instances of a class**

```
class Person {

public:
    std::string name;
    int age;

    Person(std::string name, int age) {
        this->name = name;
        this->age = age;
    }

    std::string toString() const {
        return "Name: " + name + " - Age: " + std::to_string(age);
    }
};


int main() {
    Person person(“Ronan", 30);
    std::cout << person.toString() << std::endl;

    return 0;
};
```

EXPLAIN

Class Syntax in C++



Class name

Class ends with a ;

```
class Time {

public:
      int hours;
      int minutes;

      Time( ) { … }


      string get24Format() { … }

      string get12Format() { … }
};
```

Attributes

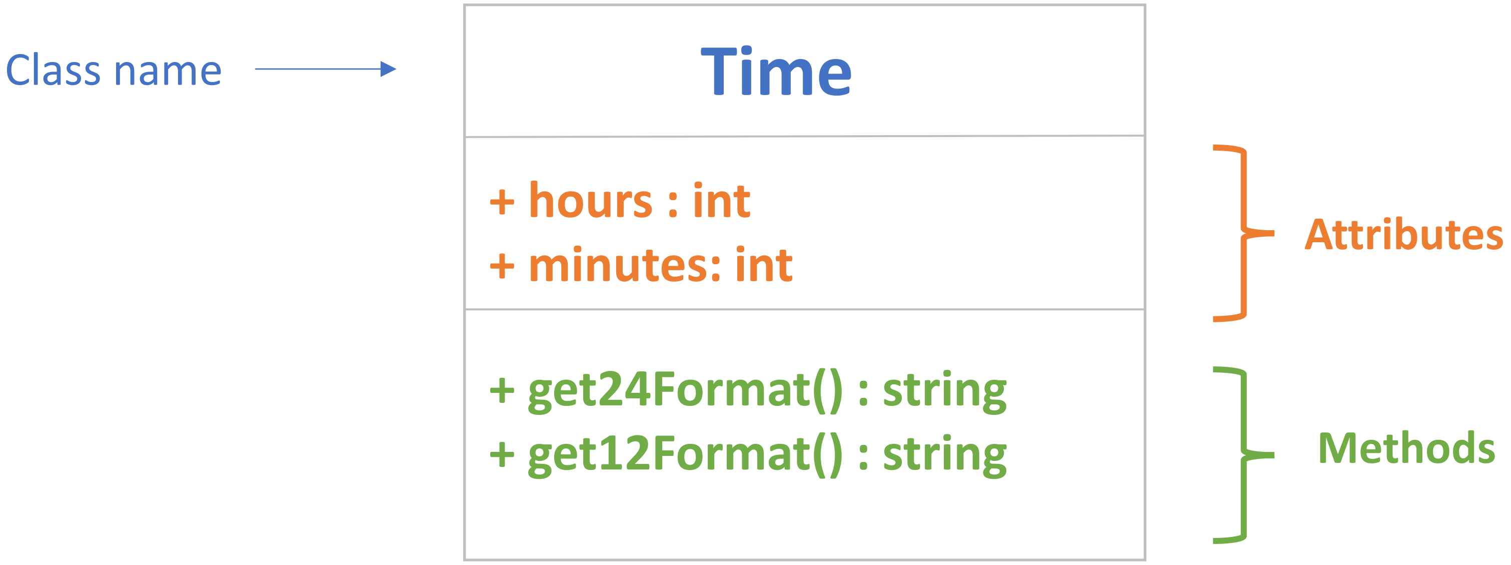
Constructors

Methods

Time class in C++

EXPLAIN

Class Representation in UML



Class name

Time

+ hours : int
+ minutes: int

+ get24Format() : string
+ get12Format() : string

Attributes

Methods

UML Diagram for Time class

EXPLAIN

Constructor

The constructor is a special method that is automatically called when an object of a class is **instantiated**.

```
class Student {
  public:
    std::string id;

    Student(std::string name) {
      this->id = name;
    }
};
```

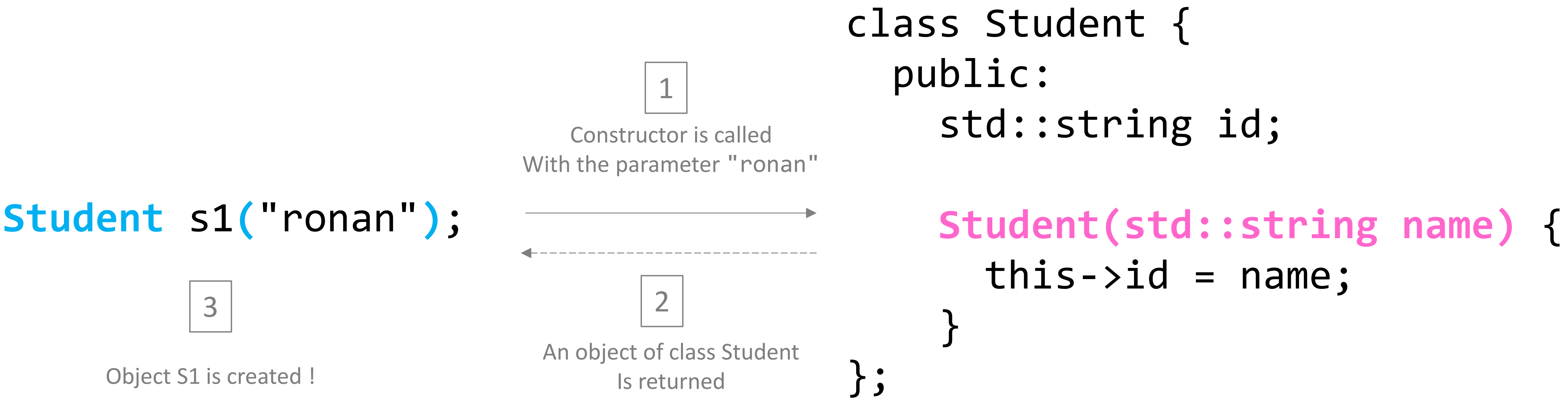
The construct can have **parameters**

The construct shall have the **name of the class**

This is used to access to the class attributes or methods

EXPLAIN

Instantiate *(create)* an object



```
Student s1("ronan");
```

1

Constructor is called
With the parameter "ronan"

2

An object of class Student
Is returned

3

Object S1 is created !

```
class Student {
  public:
    std::string id;

    Student(std::string name) {
      this->id = name;
    }
};
```

EXPLAIN

Attribute & Methods **Access**

We access to the class attributes and methods using the .

```
class Time {
 public:
      int hours;
      int minutes;

      Time( ) { … }


      string get24Format() { … }
      string get12Format() { … }
};
```

```
Time t1(13, 59);

cout << t1.minutes

cout << t1.get24Format()
```

ACTIVITY

10 MIN

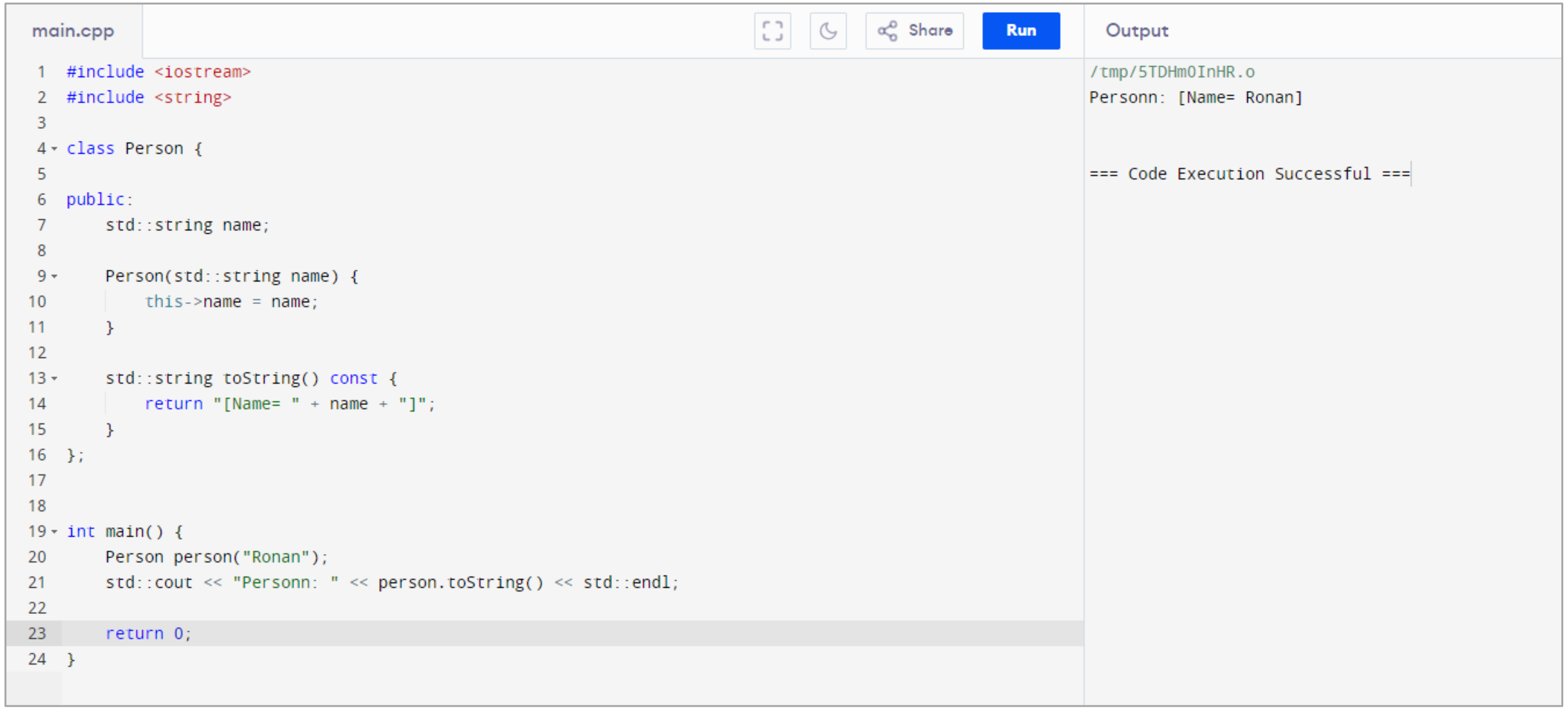
Activity 1

✓ Open the following code : https://www.programiz.com/online-compiler/1xPN9fsOPrBVY

Q1 - Add the person age (int) attribute to the class

Update the class constructor, methods, and object accordingly !

Q2 – Create another person



```cpp
#include <iostream>
#include <string>

class Person {

public:
    std::string name;

    Person(std::string name) {
        this->name = name;
    }

    std::string toString() const {
        return "[Name= " + name + "]";
    }
};


int main() {
    Person person("Ronan");
    std::cout << "Personn: " << person.toString() << std::endl;

    return 0;
}
```

```
/tmp/5TDHmOInHR.o
Personn: [Name= Ronan]


=== Code Execution Successful ===
```

EXPLAIN

Constructor by default

If there is no constructor specified in the class, the compiler considers that the class has a default constructor

```
class Student {
  public:
    int age= 30;
};

Student s1; // OK   age = 30
```

EXPLAIN

Constructor by default

If at least one constructor is specified, the compiler no longer provides a default constructor and therefore all object creations must use this(these) constructor(s).

```
class Student {
 public:
  int age= 30;

  Student(int a) { age = a;}
};

Student s1; // NOT OK

Student s2(35); // OK age = 35
```

EXPLAIN

Constructor Initializer list

The constructor provide a syntax to directly initialize the class attribute

```
class Student {
  public:
    std::string id;

    Student(std::string n) : id(name) {
     }
};
```

Attribute id is defined using the initializer list

```
class Student {
  public:
    std::string id;

    Student(std::string name) {
      this->id = name;
    }
};
```

Attribute id is defined using the constructor body

QUIZ

What will this code print?

```
class Toyota {
  public:
    std::string id= "NO ID";

    Toyota(std::string model , int year) {
      this->id = "TOYOTA - " + model + " - " + std::to_string(year);
    }
};


int main() {
    Toyota newModel("GR86 ", 2024);
    std::cout << newModel.id << std::endl;

    return 0;
}
```

1 NO ID

2 GR86, 2024

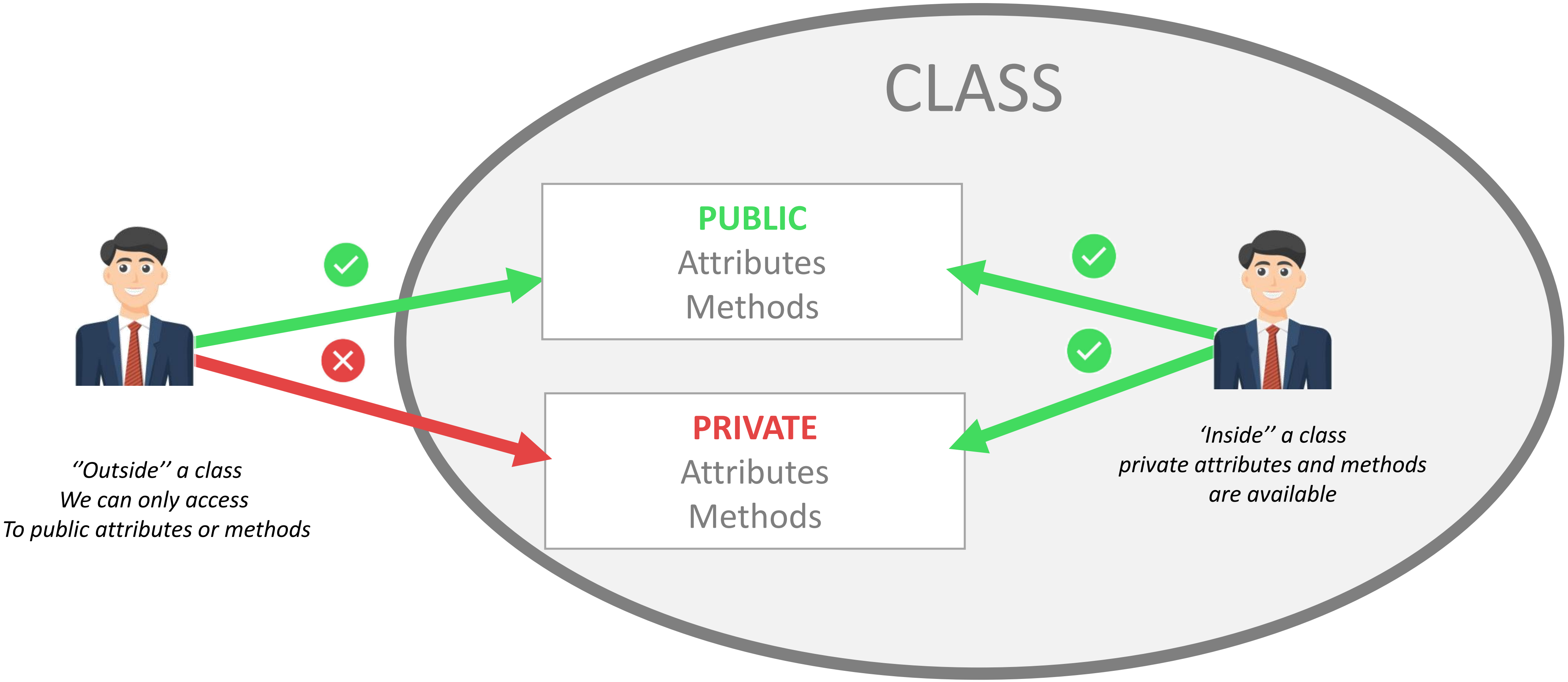
3 TOYOTA - GR86 - 2024

4 TOYOTA 2024 – NO ID

EXPLAIN

Access Modifiers

Public: Members declared as public can be accessed from outside the class
Private: Members declared as private can only be accessed within the class itself.



''Outside'' a class
We can only access
To public attributes or methods

'Inside'' a class
private attributes and methods
are available

EXPLAIN

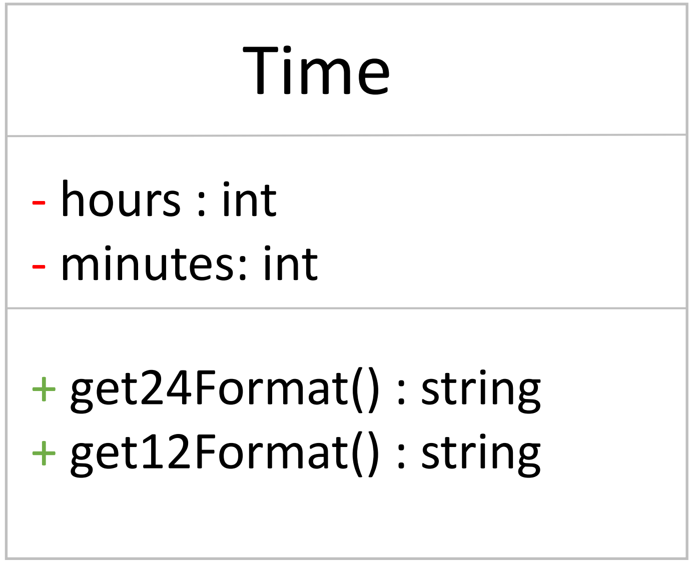
Access Modifiers

Private members

Public members

```
class Time {

private:
        int hours;
        int minutes;

public:
        Time( ) { … }


        string get24Format() { … }

        string get12Format() { … }
};
```

C++ Code for Time class



Time

- hours : int
- minutes: int

+ get24Format() : string
+ get12Format() : string

UML Diagram for Time class

EXPLAIN

Access Modifiers / Getter

When attributes are private, we can provide **getters** to read the value, without changing it

```
class Time {

private:
      int minutes;

public:
      Time(int m ) : minutes(m) {}

      int getMinutes() {
            return this-> minutes;
      }
};
```

The attribute minute cannot be changed, but can be read, using its getter
Note that the minute is returned by value

ACTIVITY

10 MIN

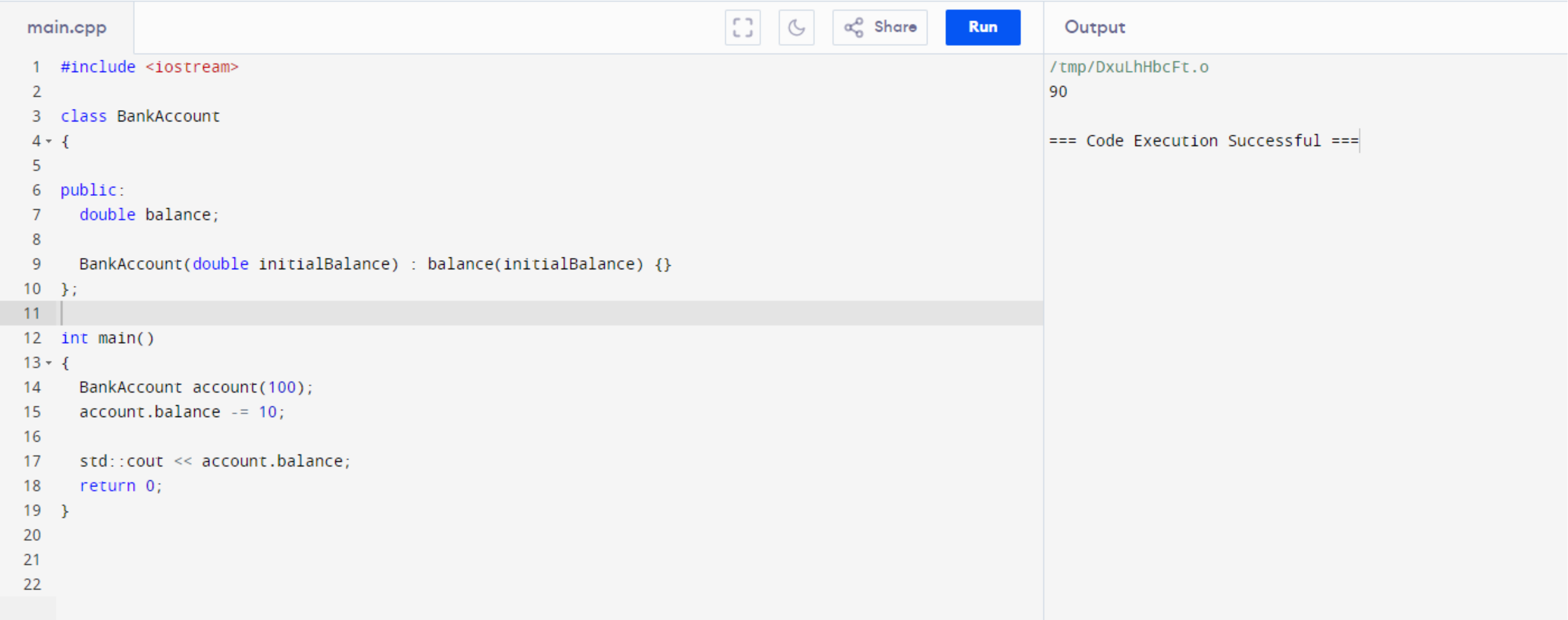
Activity 2

✓ Open the following code https://www.programiz.com/online-compiler/2VrAbKXLWGAh3

Q1 - Make the balance attribute **private**

Q2 – Instead provide a public method to change to withdraw money

Pre condition : the balance should be greater than the amount to withdraw



```cpp
#include <iostream>

class BankAccount
{

public:
  double balance;

  BankAccount(double initialBalance) : balance(initialBalance) {}
};

int main()
{
  BankAccount account(100);
  account.balance -= 10;

  std::cout << account.balance;
  return 0;
}
```

```
/tmp/DxuLhHbcFt.o
90

=== Code Execution Successful ===
```

EXPLAIN

Default parameters

It is possible to give default values to **methods** parameters

```
class Time {

private:
        int hours;
        int minutes;
        int seconds;

public:
        Time(int h, int m=60, int s =20  ) : minutes(m) {}

};

Time t1(13);                // OK, we use parameter 13, 60, 20
Time t1(13, 45);            // OK, we use parameter 13, 45, 20
Time t1(13, 45. 06);        // OK, we use parameter 13, 45, 06
```

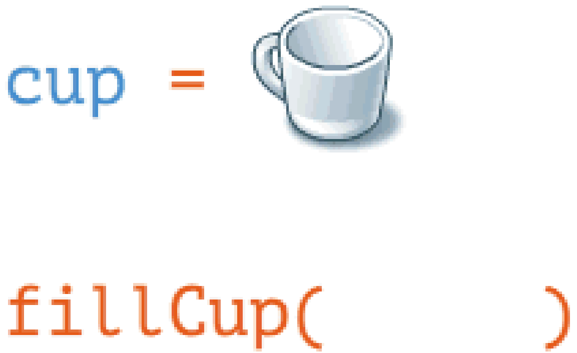
Default parameters should be the last ones in the list

EXPLAIN

Passing objects

LEARN MORE HERE

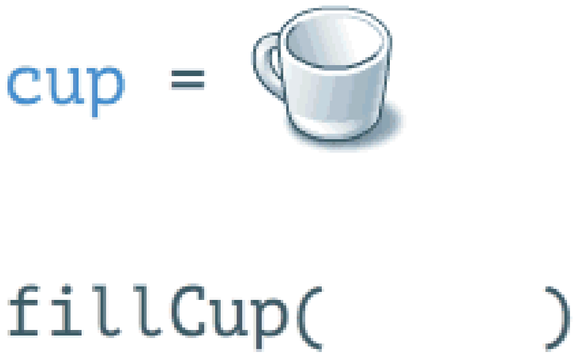
By reference



```
cup = 

fillCup(      )
```

When an object is **passed by reference**, **the memory address** of that variable is passed to the function

By value



```
cup = 

fillCup(      )
```

When an object is **passed by value**, **a copy of the object** is stored in the memory

EXPLAIN

Passing objects

Person

+ age : int

Let’s explain the difference with a class person with an age

By reference (&)

```
// This function that takes a person by reference
void setAge(Person& p, int newAge) {
    p.age = newAge;   //This changes the original object
}

…

Personn sokan(25);
setAge(sokan, 26);

// Sokan.age is now  26 !
```

By value

```
// This function that takes a person by value
void setAge(Person p, int newAge) {
    p.age = newAge;   //This changes the copy of the object
}

…

Personn sokan(25);
setAge(sokan, 26);

// Sokan.age is still 25 !
```

RESEARCH

Research Assessment

Before next session, make some researches regarding the bellow questions.

Q1 - What are the key differences between static and dynamic object instantiation in C++?

Q2 - How does memory allocation and deallocation differ between static and dynamic objects?

Q3 - What are the advantages and disadvantages of using static object instantiation versus dynamic object instantiation?

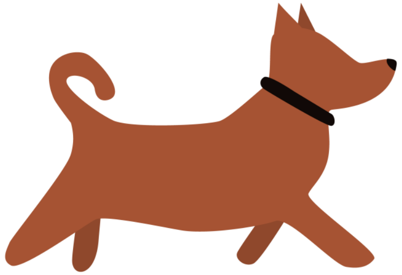
Expected outcome

- Presentation outlining the findings.
- Code examples demonstrating both static and dynamic object instantiation.
- A comparison table summarizing the differences between static and dynamic instantiation

Congratulations !

You should now master those concepts

- ✓ **Benefits of class** vs structures
- ✓ Class **attributes** & methods
- ✓ Class **instantiation** and constructors
- ✓ Class members **visibility**
- ✓ **this** as a reference to current object
- ✓ Pass objects by **value** or by **reference**
- 🔍 *(research activity)*
  Dynamic instantiation & destructors



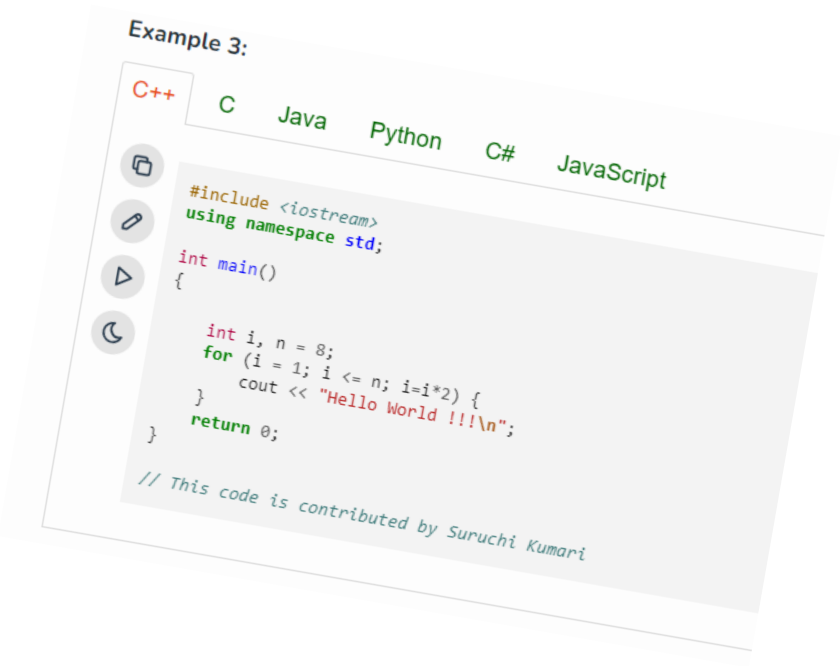
AFTER THIS SESSION

Preparation before next session

⚠ *You will need to present your research during the next session*

LEARN

1. Read this [guide about algorithm complexity]()



Example 3:

```
#include <iostream>
using namespace std;
int main()
{

    int i, n = 8;
    for (i = 1; i <= n; i=i*2) {
        cout << "Hello World !!!\n";
    }
    return 0;
}

// This code is contributed by Suruchi Kumari
```

RESEARCH

Make researches on the following points

Q1 – What does the **time complexity** means ?

Q2 – What are the **impacts** of an algorithm with a **Hight complexity** ?

3-2-1 Challenge

✓ List three things you **learned** today.

✓ List two **questions** you still have.

✓ List one aspect of the lesson or topic you **enjoyed**.