CLASSES/OBJECTS

a car is an object with fields and methods

fields → variabili
methods → funzioni
variabili, funzioni → class members

Class: is a user-defined data type that we can use in our programm, and it works as an object constructor.

es:

```
class myClass {

    public:                 // keyword is an access specifier, which specifies that members (fields and methods) of the class are
                               accessible from the outside the class.

    int num = 0;
                                fields
    string myString ;

};
```

Object: is created from a class.

We have already created the class myClass, so now we can use this to create objects.

To create an object of myClass, specify the class name, followed by the object name

To access the class attributes (myNum and myString), use the . syntax on the object.

es:

```
class myClass {

    public:

    int num = 0;

    string myString ;

};

int main () {

myClass myObj;        // create an object of myClass
```

```
myObj.myNum = 15;              // access fields and set values

myObj.myString = "some text";

cout << myObj.myNum << endl;   // print fields and values

cout << myObj.myString;

return 0;

}
```

Class Methods: functions that belongs to a class.

There are two ways to define functions that belongs to a class:

* Inside class definition

* Outside class definition

Now, we define a function inside the class. [You access methods just like you access fields: by creating an object of the class and using the .]

ex:

```
class myClass {

        public:

        void myMethod () {

                cout << "Hello Word";

                }

        };

int main () {

myClass myObj;

myObj.myMethod ();

return 0;

}
```

You can also add paramaters at the methods:

```
#include <iostream>

using namespace std;

class Car {

        public:

        int speed (int maxSpeed) {

                                return maxSpeed;

                                }

};

int main () {

car myCar;

cout << myCar.Speed (200);

return 0;

}
```

Constructors: is a special method that is automatically called when an object of a class is created

To create a constructor, use the same name as the class, followed by the parentheses ():

ex:

```
class myClass {

        public :

        myClass () {                    //constructor

                cout << "Hello world";

                }

};
```

```
int main(){

myClass myObj;          //create an object of myClass (this will call the constructor)

return 0;

}
```

n.b. The constructor has the same name as the class, it's always public and does not return any value!

Constructor can also take parameters, which can be useful for setting initial values for fields.

es:

```
class Car {

        public:

        string model;

        string brand;

        int    year;

        Car (string x, string y, int z) {

                brand = x;

                model = y;

                year = z;

                }

};

int main () {

Car carObj1 ("BMW", "X5", 1999);                    //create car objects and call the constructor with different values

Car carObj2 ("Ford", "Mustang", 1969);

cout << carObj1.brand << " " << carObj1.model << " " << carObj1.year << endl;

cout << carObj2.brand << " " << carObj2.model << " " << carObj2.year << endl;

return 0; }
```

Access specifiers: define how the members of a class can be accessed

In C++, there are three access specifiers:

- public: members are accessible from outside the class
- private: members cannot be accessed (or viewed) from outside the class
- protected: members cannot be accessed from o