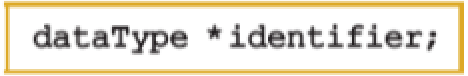
Pointer variables store memory address.
The value of a pointer variable is an address.

In C++, you declare a pointer variable by using the asterisk symbol (*) between the datatype and the variable name



```
dataType *identifier;
```

Before discussing how pointers work, let us make the following observations. The statement:

```
int *p;
```

is equivalent to the statement:

```
int*  p;
```

which is equivalent to the statement:

```
int * p;
```

Thus, the character `*` can appear anywhere between the data type name and the variable name.

Now, consider the following statement:

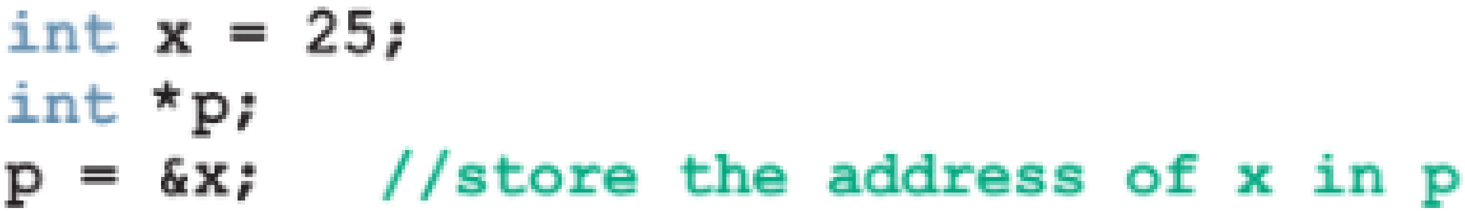
```
int* p, q;
```

In this statement, only `p` is the pointer variable, not `q`. Here, `q` is an `int` variable. To avoid confusion, we prefer to attach the character `*` to the variable name. So the preceding statement is written as:

```
int *p, q;
```

Of course, the statement:

```
int *p, *q;
```



```
int x = 25;
int *p;
p = &x;    //store the address of x in p
```

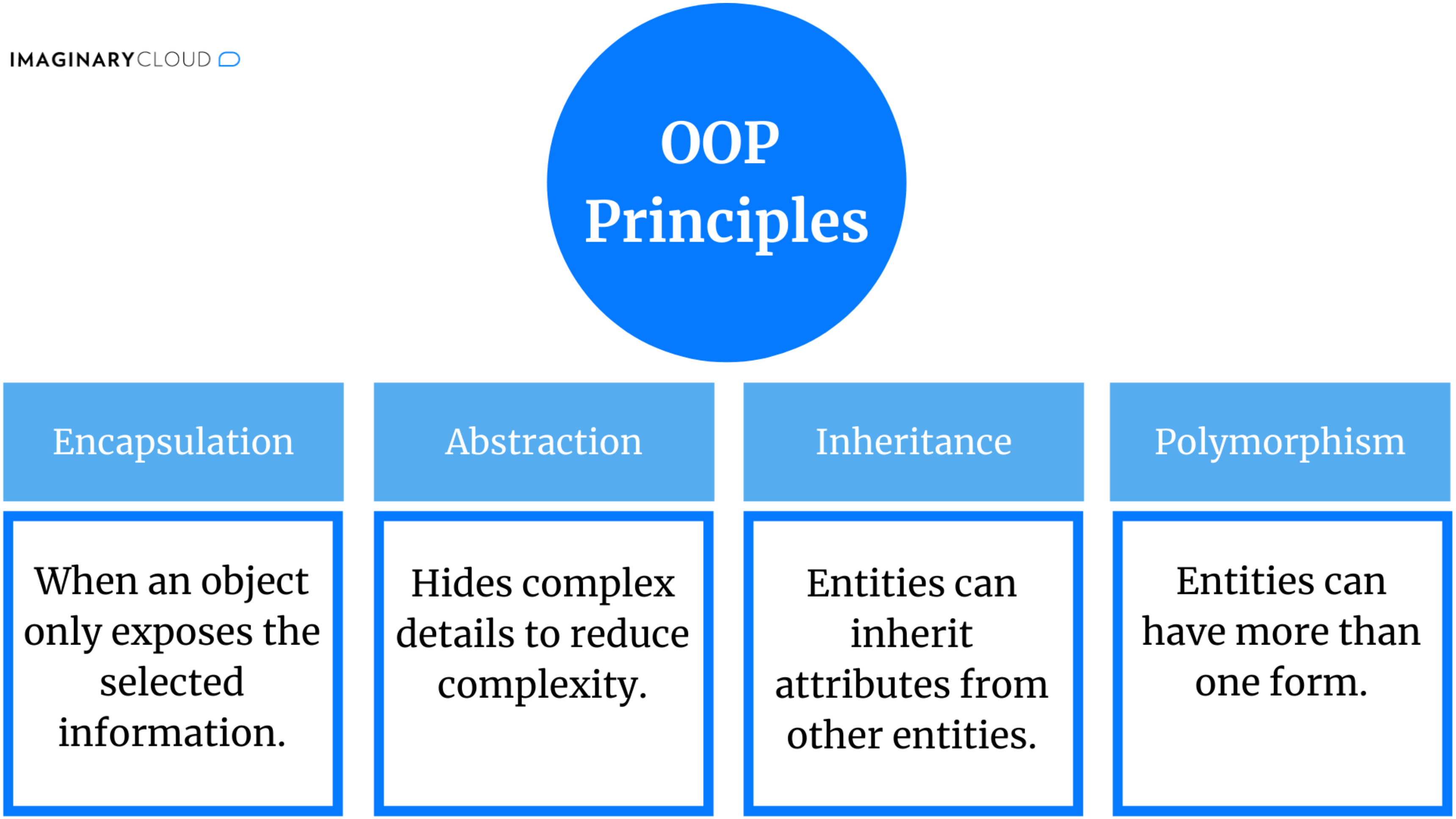
```
delete p;
```

EXAMPLE 14-3

The following program illustrates how pointer variables work:

```
//Chapter 14: Example 14-3

#include <iostream>

using namespace std;

int main()
{
    int *p;
    int x = 37;

    cout << "Line 1: x = " << x << endl;              //Line 1

    p = &x;                                           //Line 2

    cout << "Line 3: *p = " << *p
         << ", x = " << x << endl;                    //Line 3

    *p = 58;                                          //Line 4

    cout << "Line 5: *p = " << *p
         << ", x = " << x << endl;                    //Line 5

    cout << "Line 6: Address of p = " << &p << endl;  //Line 6

    cout << "Line 7: Value of p = " << p << endl;     //Line 7

    cout << "Line 8: Value of the memory location "
         << "pointed to by *p = " << *p << endl;      //Line 8
    cout << "Line 9: Address of x = " << &x << endl;  //Line 9
    cout << "Line 10: Value of x = " << x << endl;    //Line 10

    return 0;
}
```

IMAGINARYCLOUD



OOP Principles

Encapsulation

When an object only exposes the selected information.

Abstraction

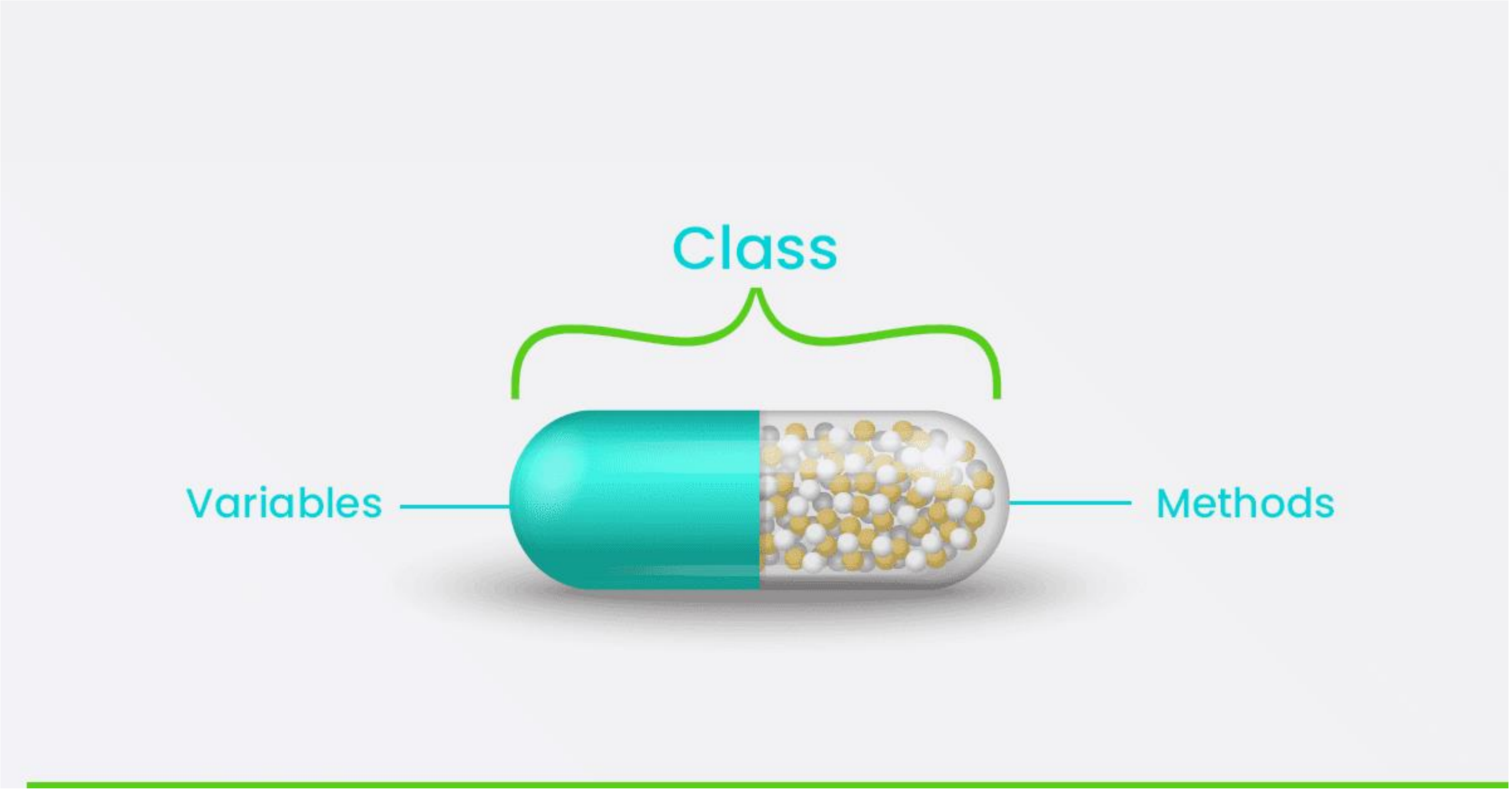
Hides complex details to reduce complexity.

Inheritance

Entities can inherit attributes from other entities.

Polymorphism

Entities can have more than one form.



Class

Variables

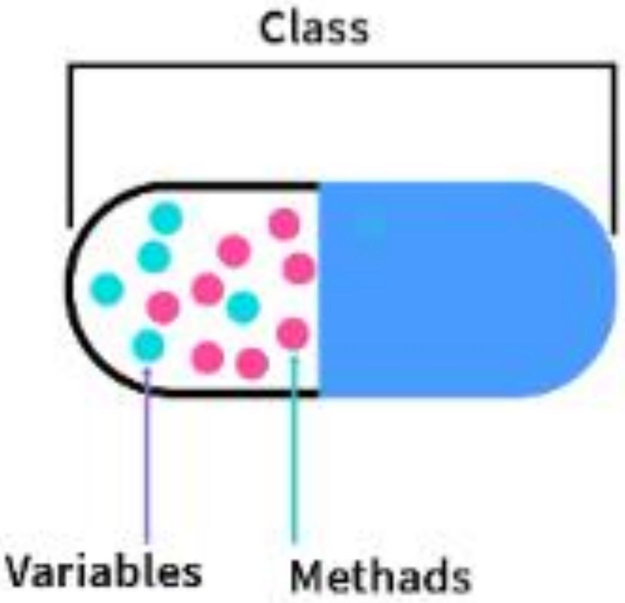
Methods

- Encapsulation, in general, is nothing but a fancy word for packaging or enclosing things of interest into one entity.
- The most common example of such a unit would be a class/object.

Person class and its objects

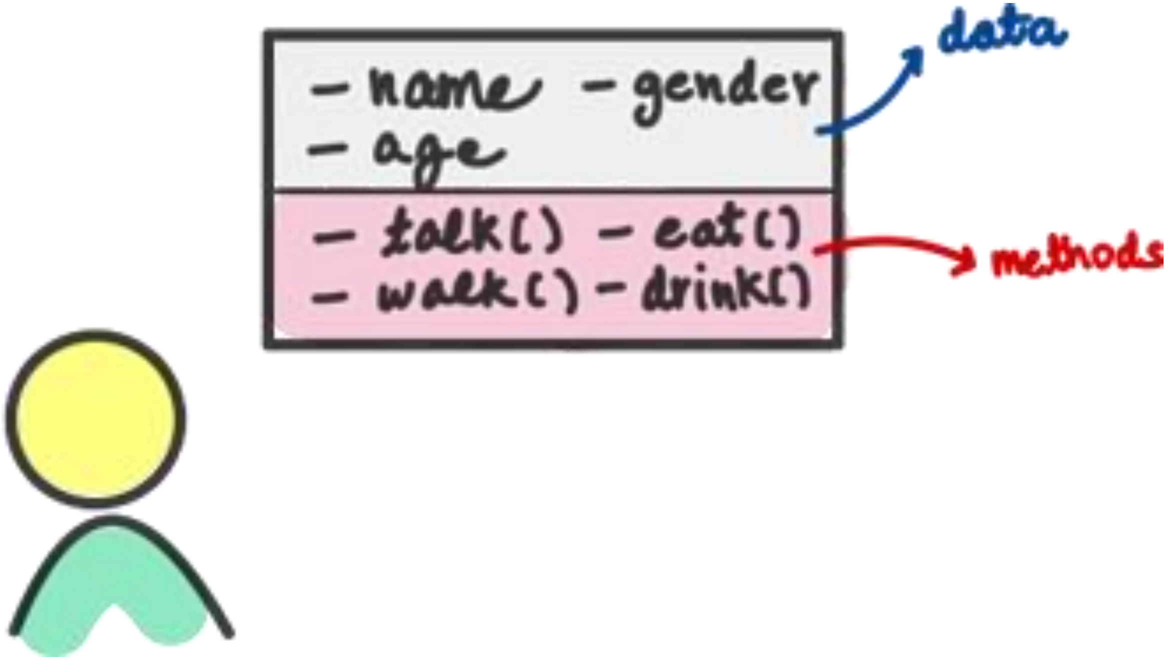
Objects of the animal class

```
Classs
{
    data members
          +
  methods (behavior)
}
```



human being into a programmable entity

In that case, the data would distinguish one human from another, and its methods would define possible operations (behavior/actions).

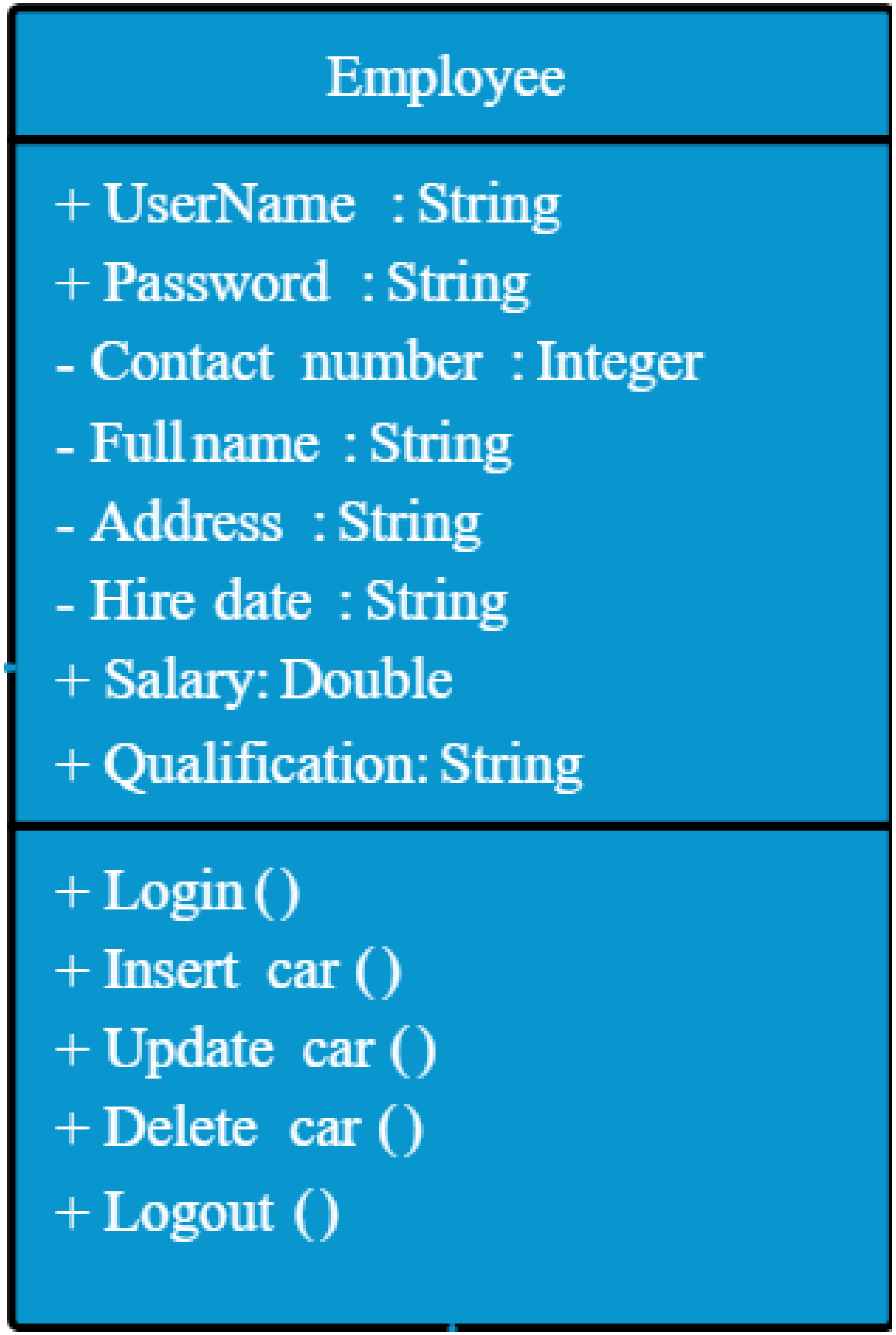


How to define class in C++

```
class ClassName {
 // properties or fields
 // methods or functions
 };
```

How to define Objects in C++

```
ClassName objectName =ClassName();
```

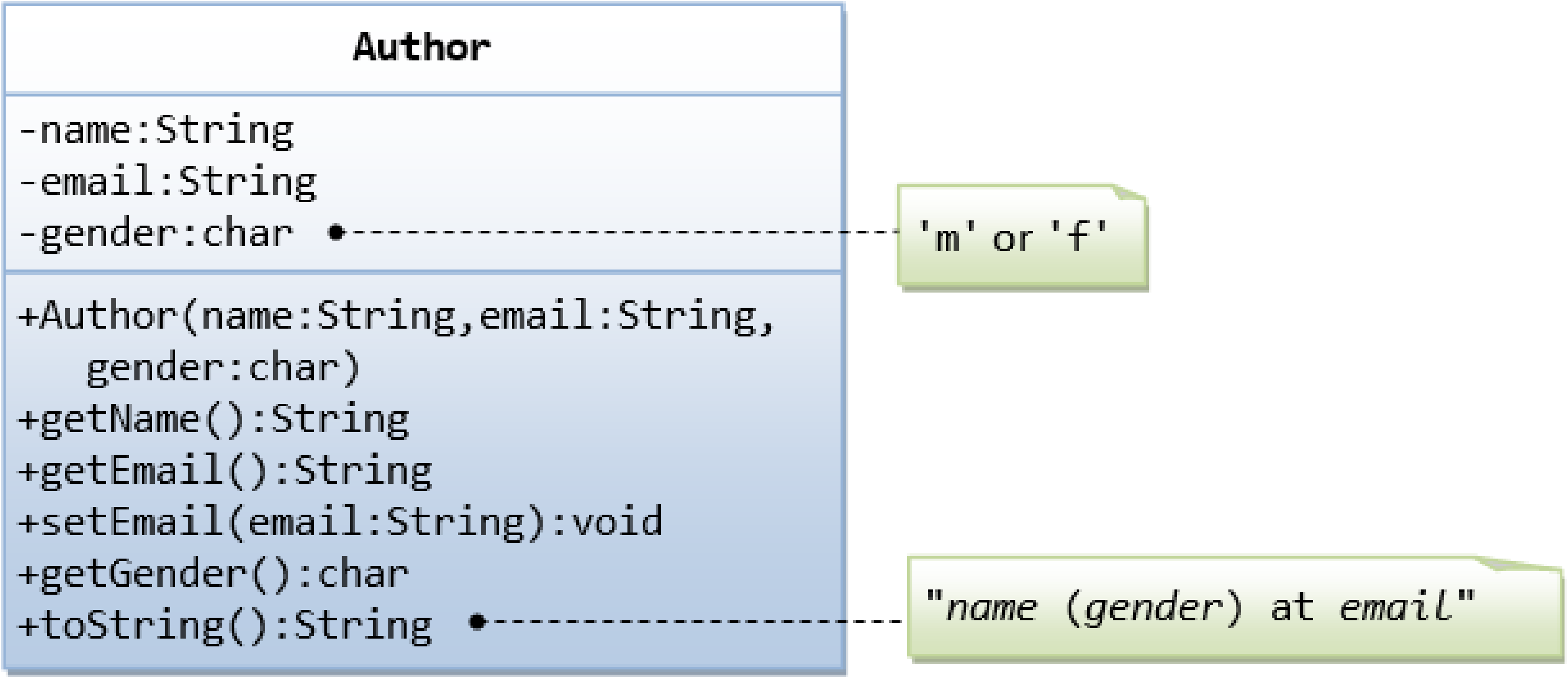


Employee

+ UserName : String
+ Password : String
- Contact number : Integer
- Full name : String
- Address : String
- Hire date : String
+ Salary: Double
+ Qualification: String

+ Login ()
+ Insert car ()
+ Update car ()
+ Delete car ()
+ Logout ()

Home work

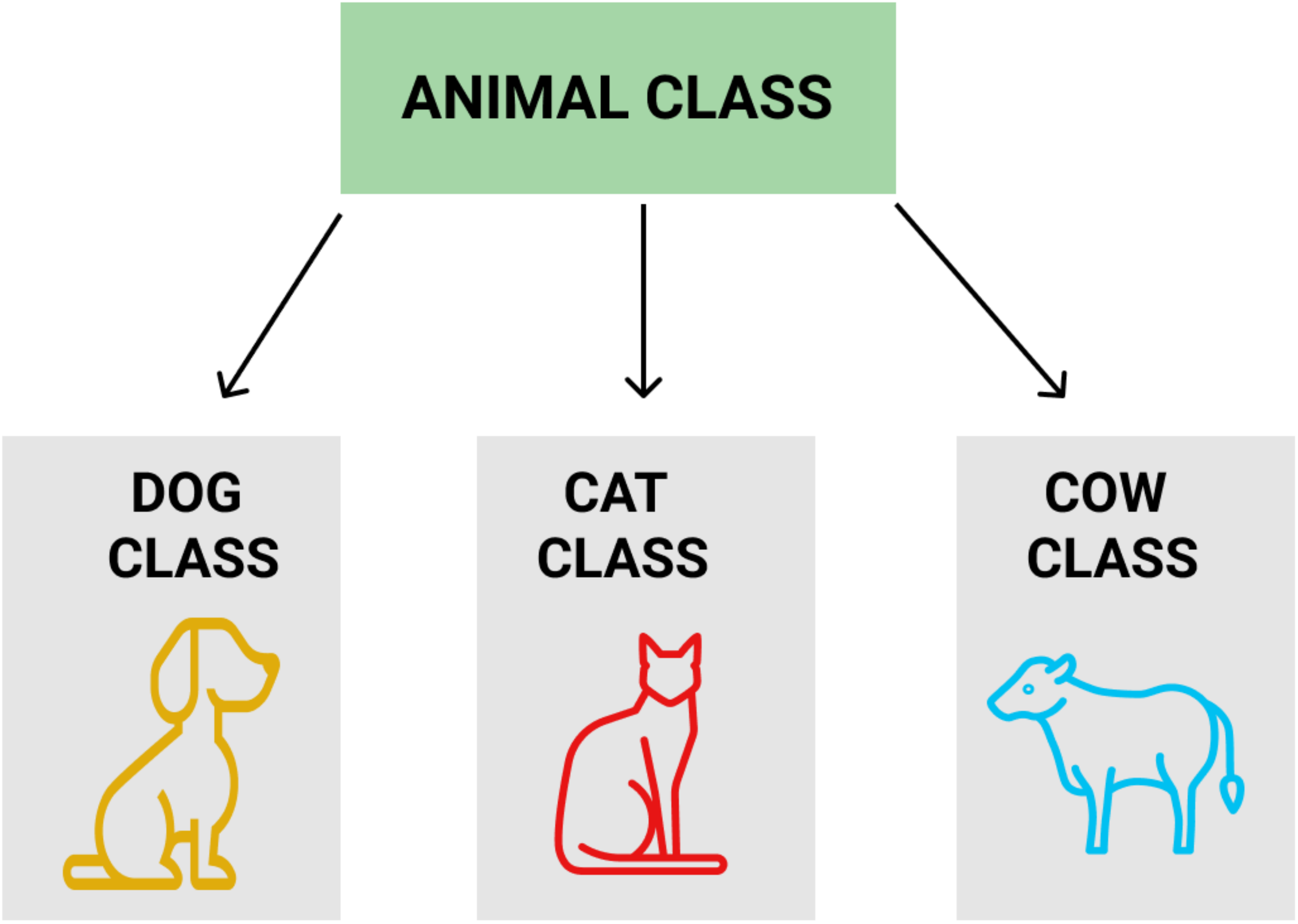


```
Author
-name:String
-email:String
-gender:char
+Author(name:String,email:String,
   gender:char)
+getName():String
+getEmail():String
+setEmail(email:String):void
+getGender():char
+toString():String
```

'm' or 'f'

"*name* (*gender*) at *email*"

INHERITANCE IN DART

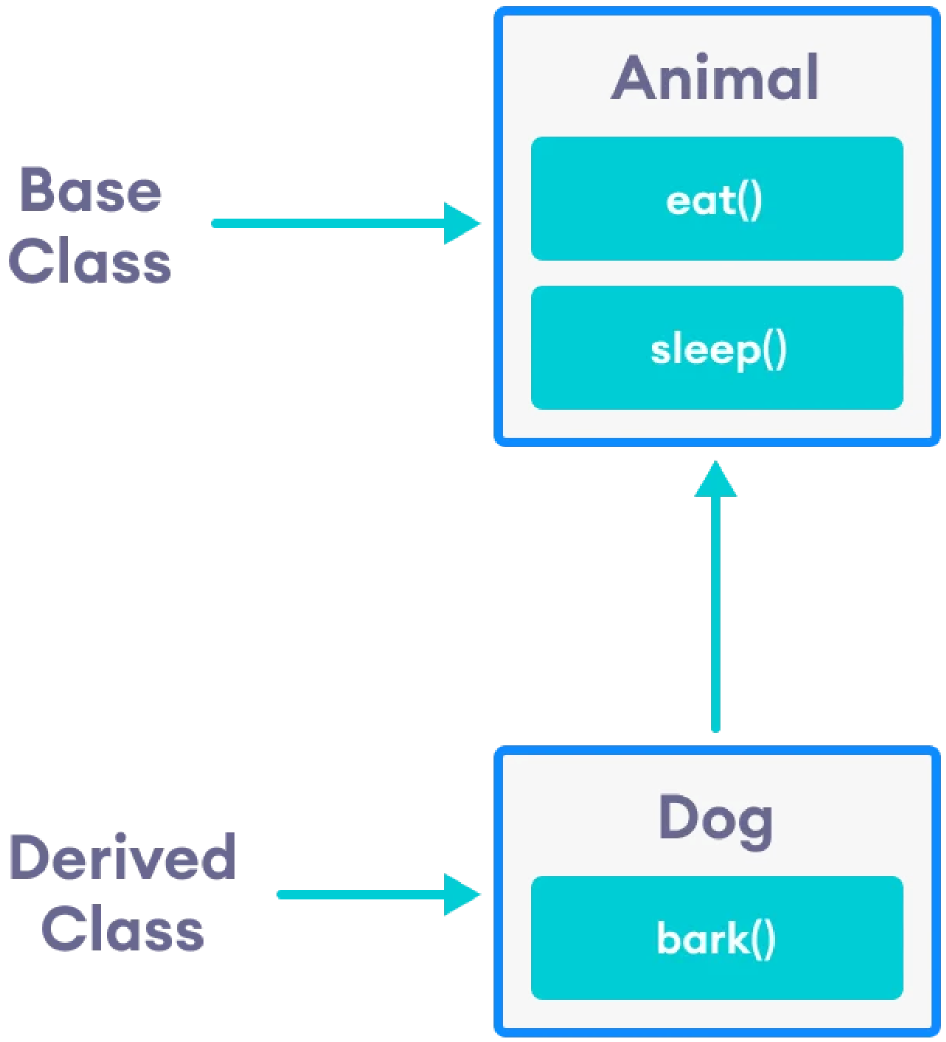


ANIMAL CLASS

DOG CLASS

CAT CLASS

COW CLASS



Base Class

Animal

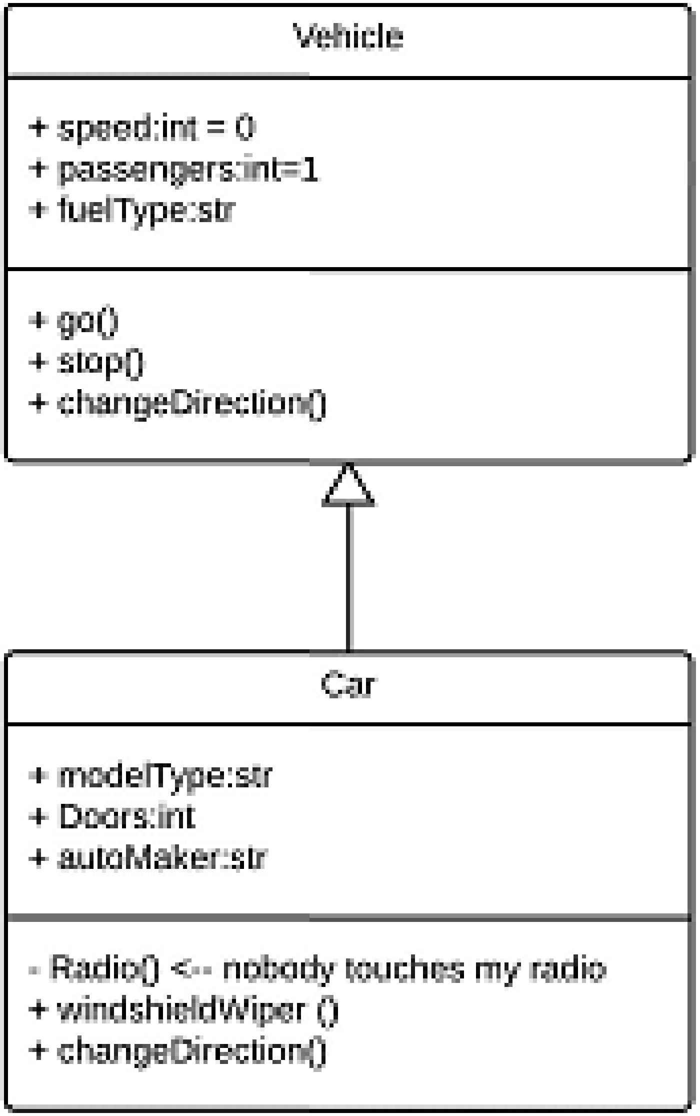
eat()

sleep()

Derived Class

Dog

bark()



Vehicle

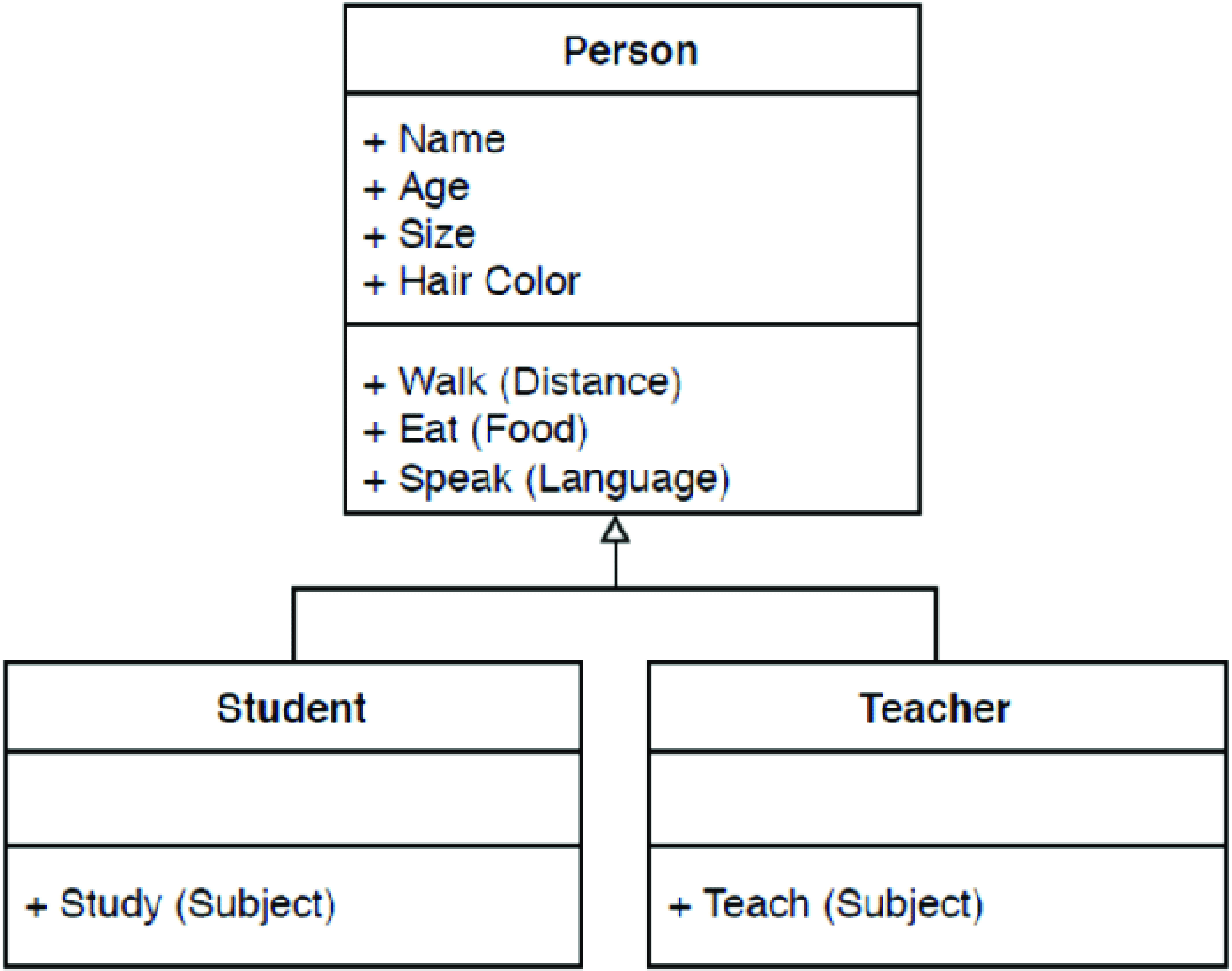
+ speed:int = 0
+ passengers:int=1
+ fuelType:str

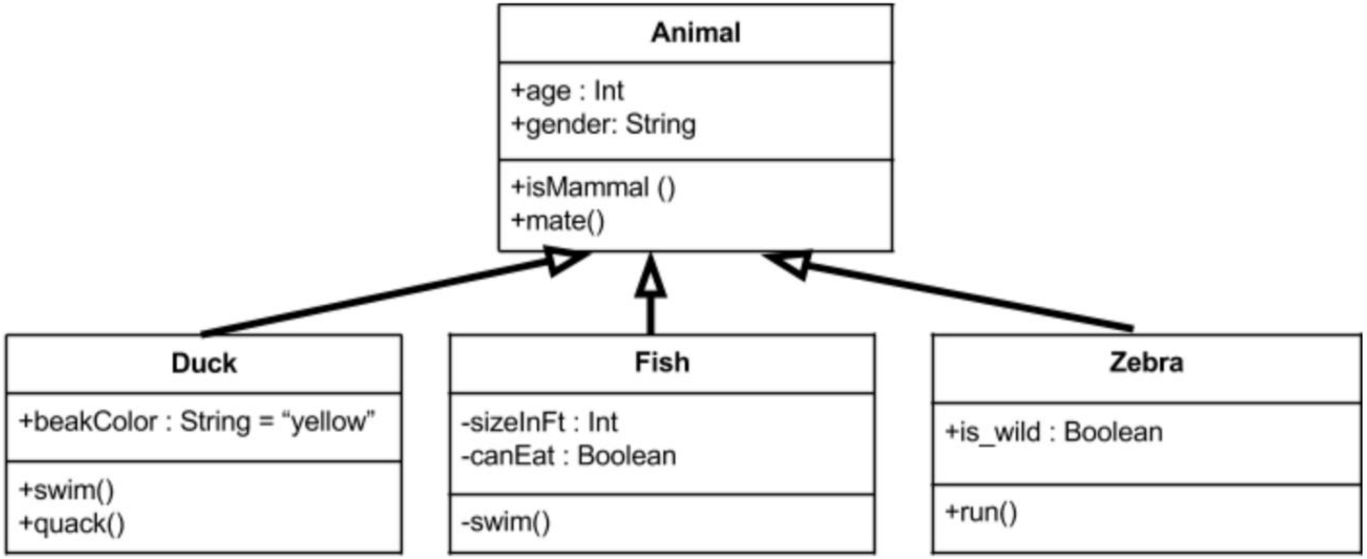
+ go()
+ stop()
+ changeDirection()

Car

+ modelType:str
+ Doors:int
+ autoMaker:str

- Radio() <-- nobody touches my radio
+ windshieldWiper ()
+ changeDirection()


Person
+ Name
+ Age
+ Size
+ Hair Color
+ Walk (Distance)
+ Eat (Food)
+ Speak (Language)
Student
+ Study (Subject)
Teacher
+ Teach (Subject)



Animal

+age : Int
+gender: String

+isMammal ()
+mate()

Duck

+beakColor : String = “yellow”

+swim()
+quack()

Fish

-sizeInFt : Int
-canEat : Boolean

-swim()

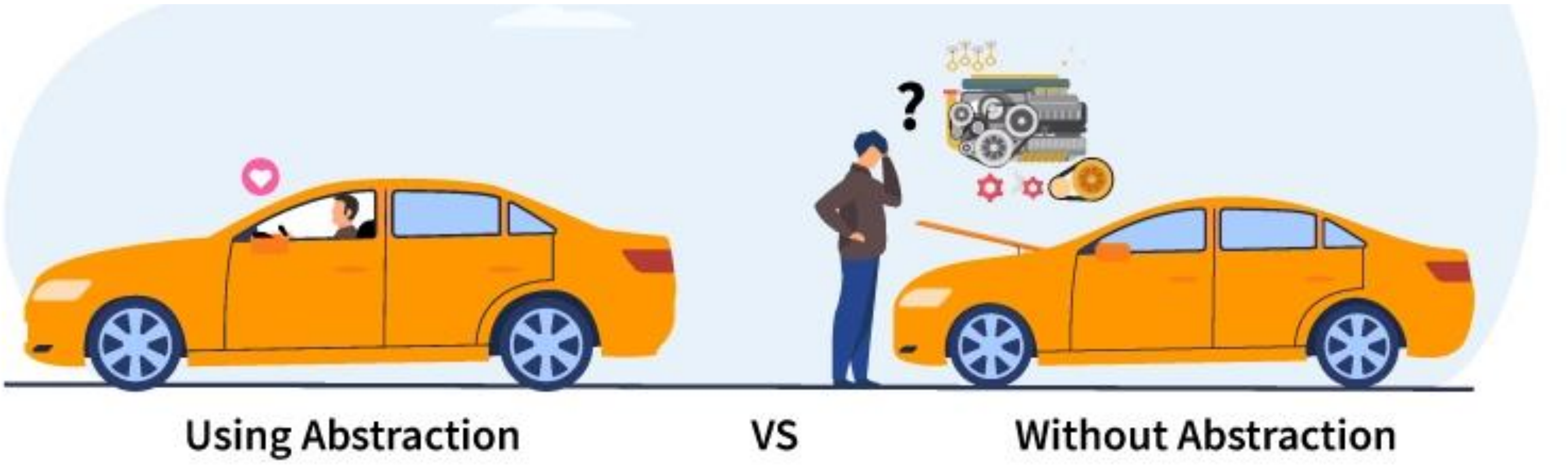
Zebra

+is_wild : Boolean

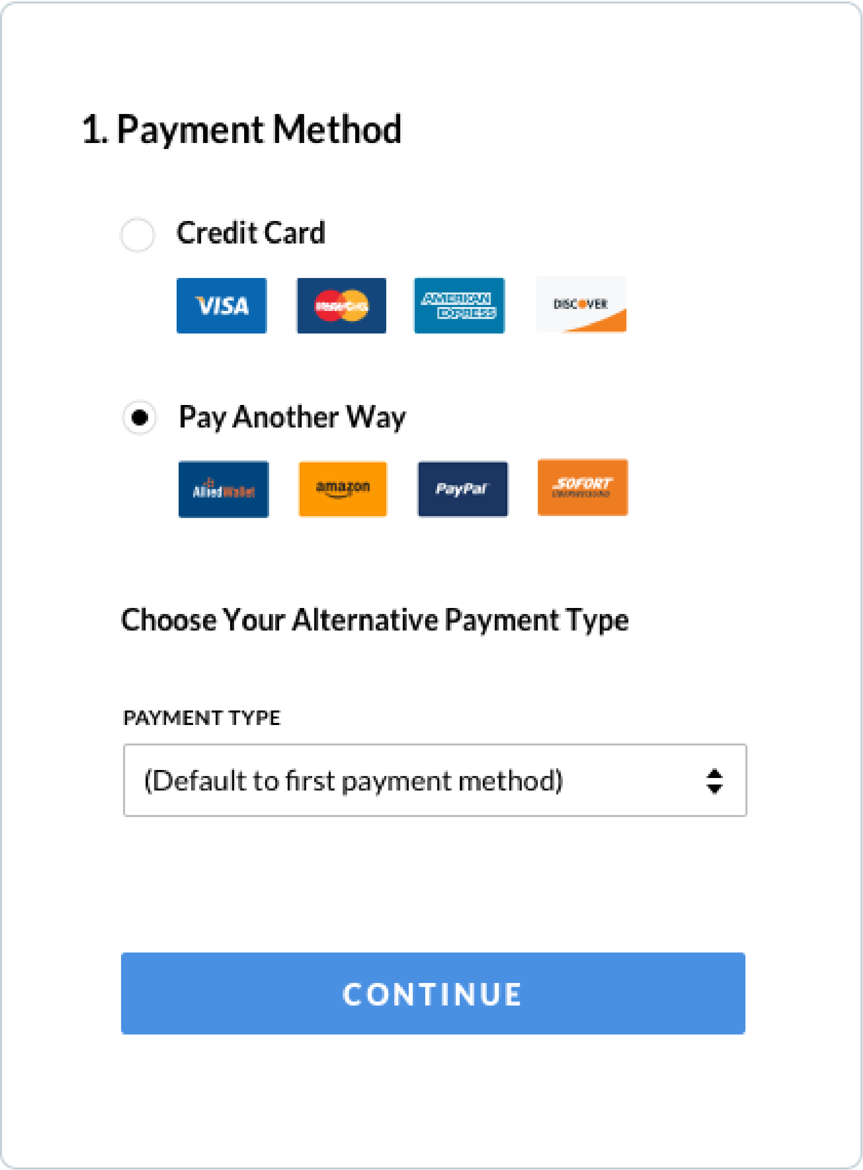
+run()

Abstraction

Abstraction is a way to hide complexities and give a simple user interface to the user.



When do we need abstraction?



1. Payment Method

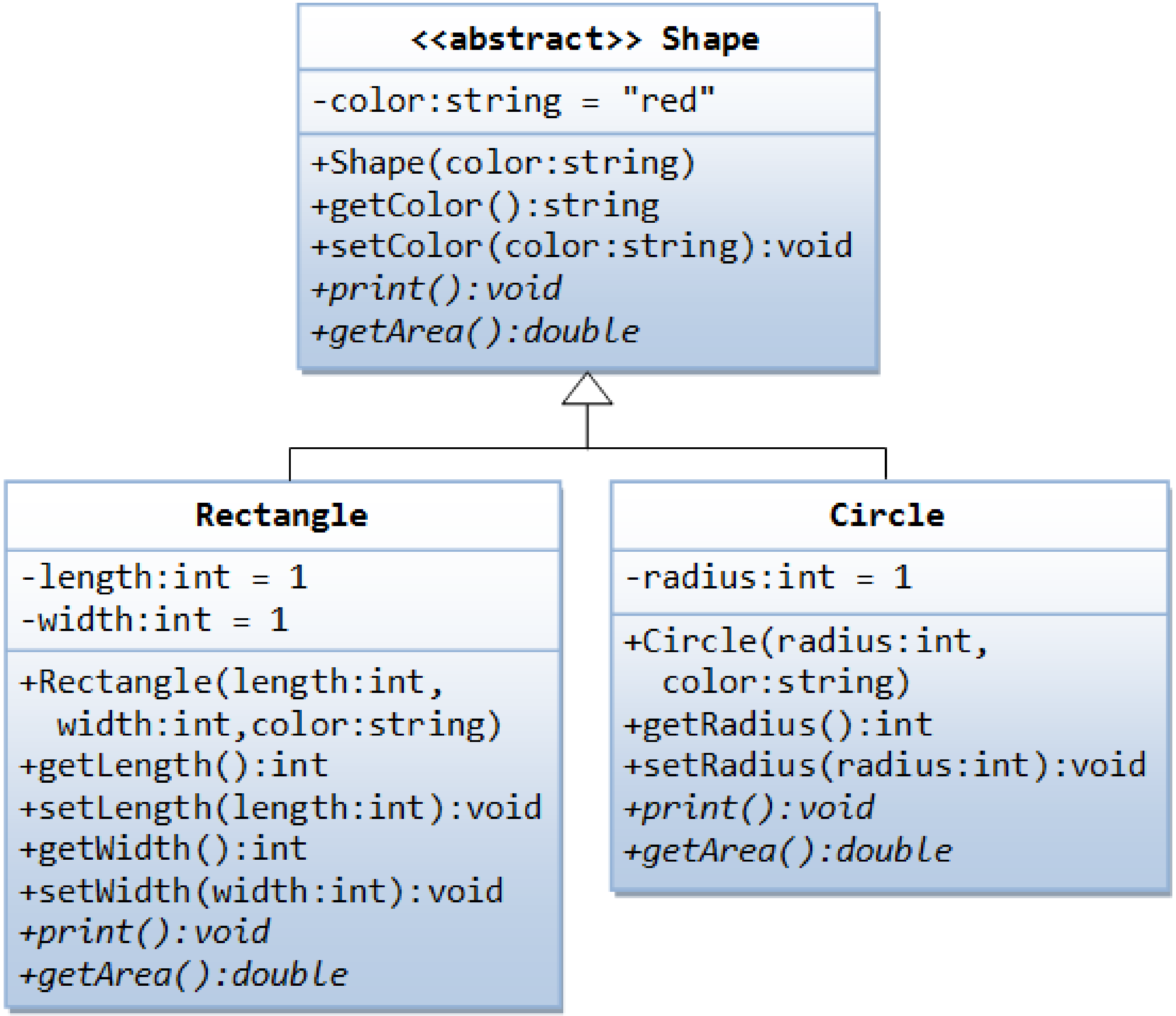
○ Credit Card

● Pay Another Way

Choose Your Alternative Payment Type

PAYMENT TYPE

(Default to first payment method)



<<abstract>> Shape

-color:string = "red"

+Shape(color:string)
+getColor():string
+setColor(color:string):void
+print():void
+getArea():double

Rectangle

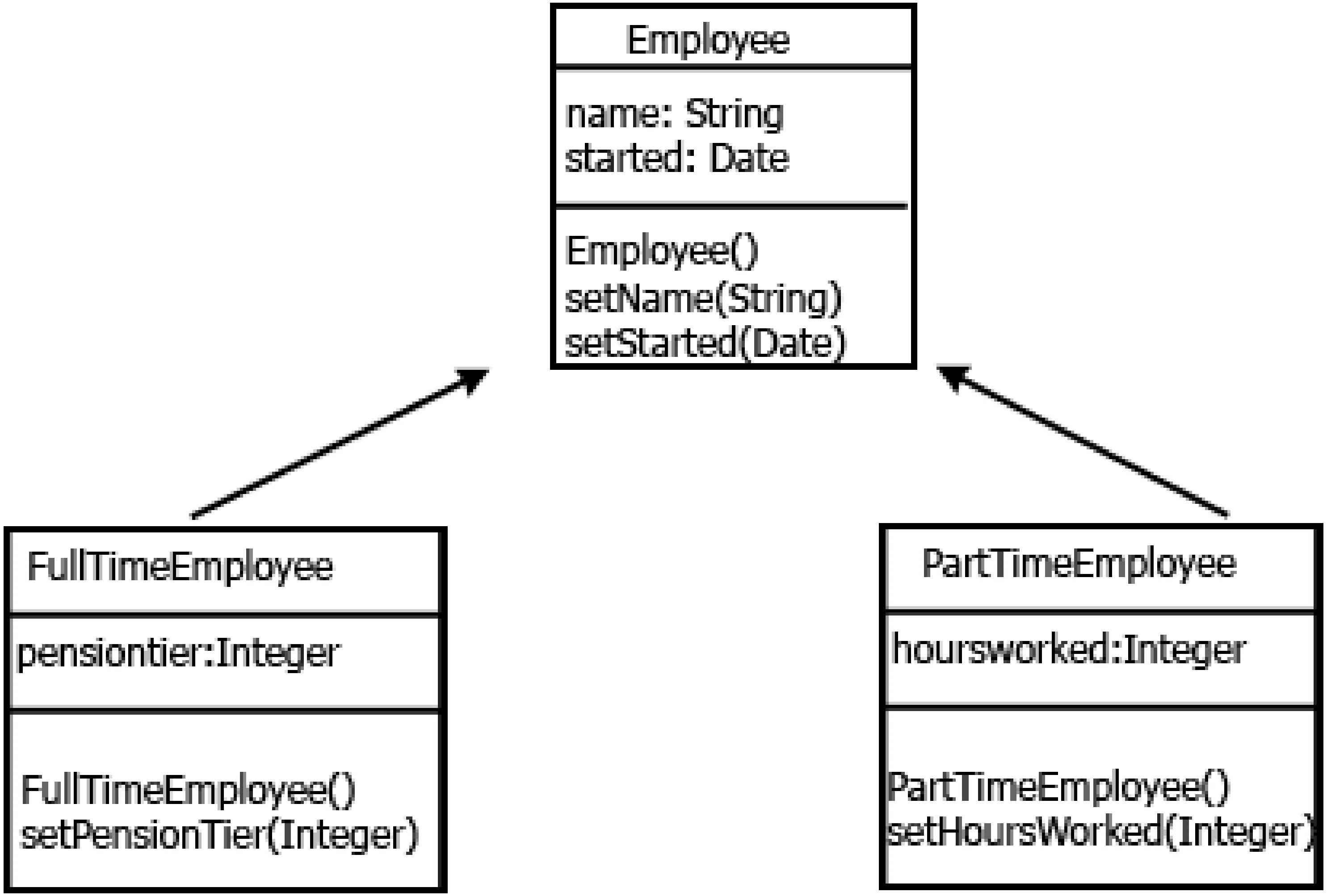
-length:int = 1
-width:int = 1

+Rectangle(length:int,
width:int,color:string)
+getLength():int
+setLength(length:int):void
+getWidth():int
+setWidth(width:int):void
+print():void
+getArea():double

Circle

-radius:int = 1

+Circle(radius:int,
color:string)
+getRadius():int
+setRadius(radius:int):void
+print():void
+getArea():double



Employee

name: String
started: Date

Employee()
setName(String)
setStarted(Date)

FullTimeEmployee

pensiontier:Integer

FullTimeEmployee()
setPensionTier(Integer)

PartTimeEmployee

hoursworked:Integer

PartTimeEmployee()
setHoursWorked(Integer)