Dart Getters and Setters

Getters and setters are special class methods that is used to initialize and retrieve the values of class fields respectively. The setter method is used to set or initialize respective class fields, while the getter method is used to retrieve respective class fields. All classes have default getter and setter method associated with it. However, you are free to override the default ones by implementing the getter and setter method explicitly.

Defining a getter

The getters are defined using the get keyword with no parameters and returns a value.

Syntax:-

```
return_type get field_name
{
}
```

Defining a setter

The setters are defined using the set keyword with one parameter and no return value.

Syntax:-

```
set field_name
{
}
```

Example:-

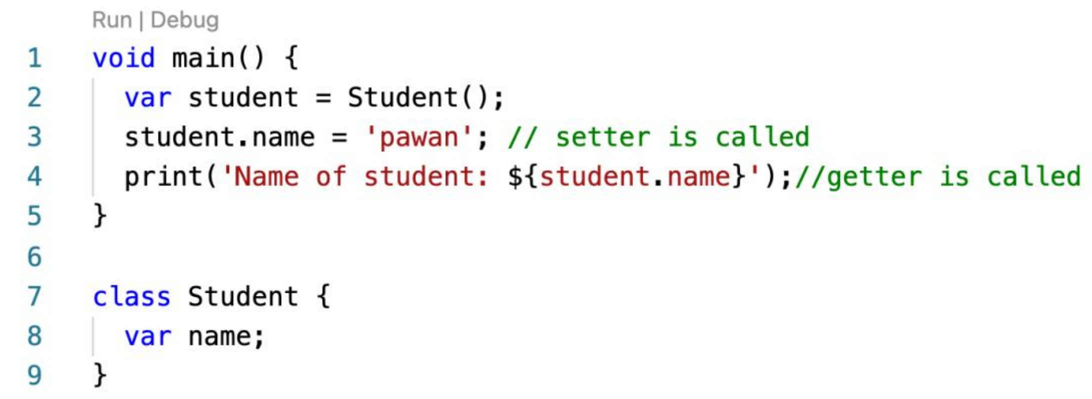
The following example shows how you can use **getters** and **setters** in a Dart class –

```
class Employee {
  String empName;
  int empAge;
  int empSalary;

  String get emp_name {
    return empName;
  }

  void set emp_name(String name) {
    this.empName = name;
  }

  void set emp_age(int age) {
    if(age<= 18) {
      print("Employee Age should be greater than 18 Years.");
    }  else {
      this.empAge = age;
    }
  }

  int get emp_age {
    return empAge;
  }

 void set emp_salary(int salary) {
    if(salary<= 0) {
      print("Salary should be greater than 0");
    }  else {
      this.empSalary = salary;
    }
  }

  int get emp_salary {
    return empSalary;
  }

}
void main() {
  Employee emp = new Employee();
  emp.emp_name = 'John';
  emp.emp_age = 25;
  emp.emp_salary = 25000;
  print("W3Adda - Dart Getters and Setters Example.");
  print("Employee's Name Is : ${emp.emp_name}");
  print("Employee's Age Is : ${emp.emp_age}");
  print("Employee's Salary Is : ${emp.emp_salary}");
}
```

There are two ways to make getter and setter in dart:

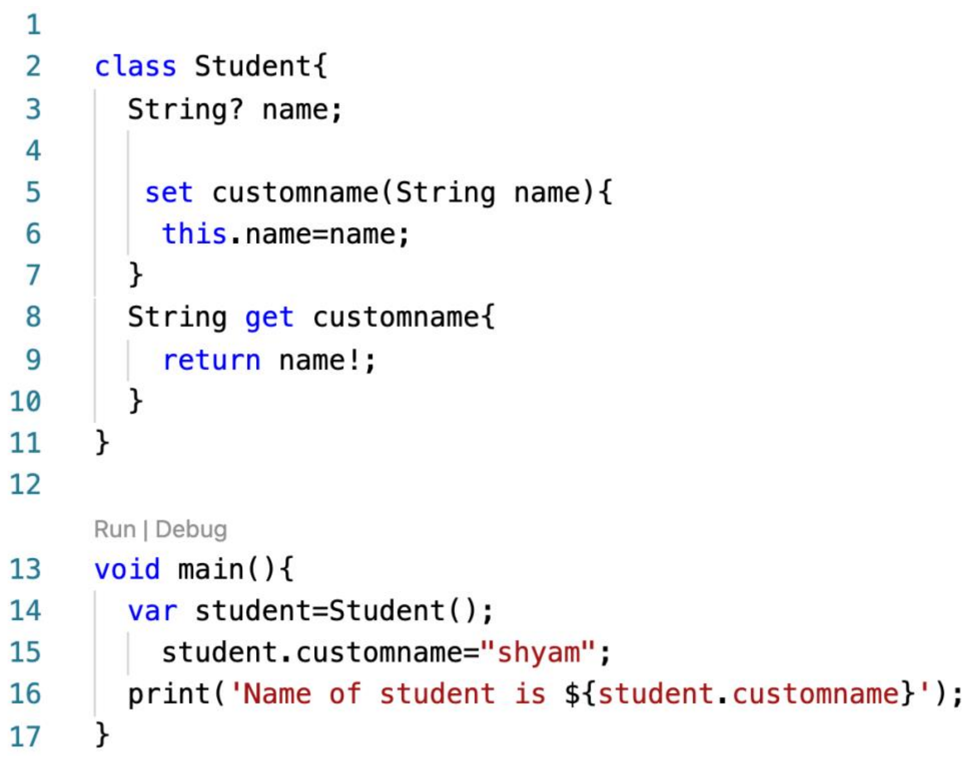
1. **Default getter and setter (implicit)**
2. **Custom getter and setter (explicit)**

<u>Default getter and setter:</u>



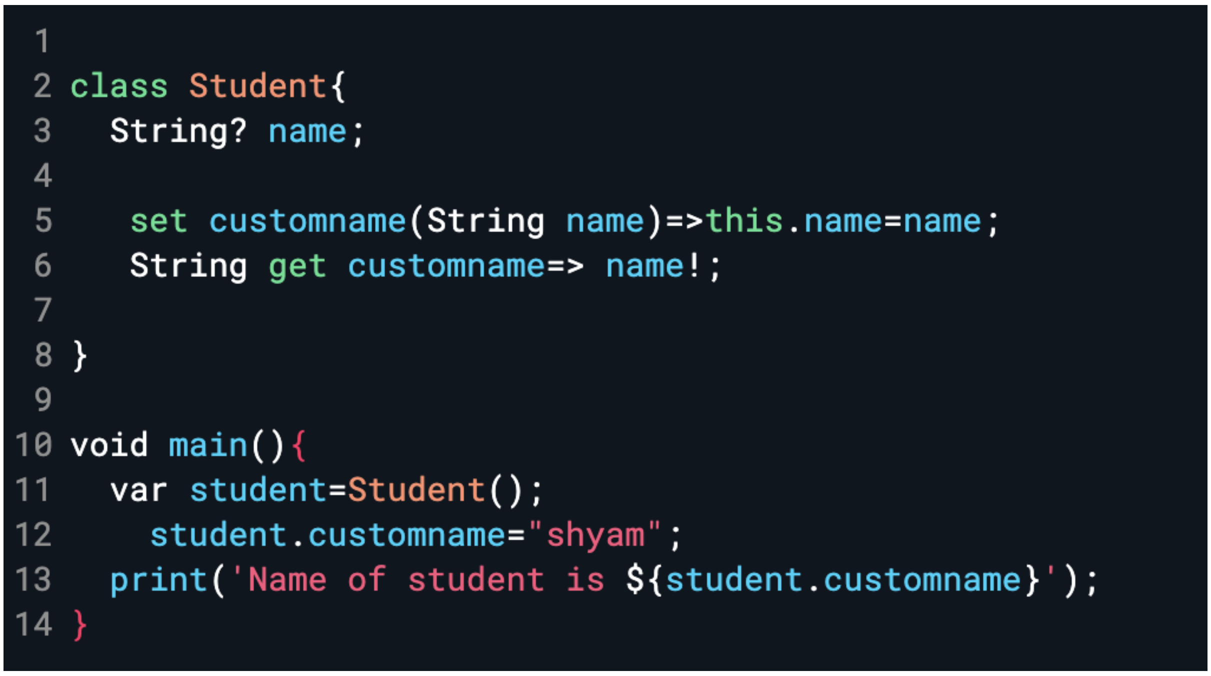
```dart
void main() {
  var student = Student();
  student.name = 'pawan'; // setter is called
  print('Name of student: ${student.name}');//getter is called
}

class Student {
  var name;
}
```

<u>Custom getter and setter:</u>



```dart

class Student{
  String? name;

   set customname(String name){
    this.name=name;
  }
  String get customname{
    return name!;
  }
}

void main(){
  var student=Student();
    student.customname="shyam";
  print('Name of student is ${student.customname}');
}
```

Now, you can trim down the custom setter and getter like this:



```

class Student{
  String? name;

   set customname(String name)=>this.name=name;
   String get customname=> name!;

}

void main(){
  var student=Student();
    student.customname="shyam";
  print('Name of student is ${student.customname}');
}
```

Note:

- Remember when defining custom setter and getter, name should be same.
- Return type in setter method is optional.
- Return type in getter is compulsory.
- If instance is nullable don't forget to check null safety.

Dart Overriding Method:

When we declare the same method in the subclass, which is previously defined in the superclass is known as the method overriding. The subclass can define the same method by providing its own implementation, which is already exists in the superclass. The method in the superclass is called **method overridden,** and method in the subclass is **called method overriding**. Let's understand the method overriding in the following example.

Method Overriding Example

We define two classes; first, is a subclass called **Human,** and the second is a superclass **Boy**. The Boy subclass inherits the Human superclass. The same method **void showInfo()** in both classes is defined with the different implementation. The subclass has its own definition of the void **showInfo().** Let's have a look at the following code snippet.

Example -

```
class Human{
  //Overridden method
   void run()
  {
     print("Human is running");
  }
}
class Man extends Human{
  //Overriding method
   void run(){
     print("Boy is running");
  }
}
void main(){
     Man m = new Man();
     //This will call the child class version of run()
     m.run();
}
```