Extending a Class

In this lesson, we will look at an example of inheritance.

We'll cover the following

Let’s look at an example to better understand inheritance.

Syntax

In Dart, we use `extends` to create a subclass.

```
class subclassName extends superclassName
```

Implementation

Let’s look at an example of a vending machine. In a food vending machine, we have several types of products such as beverages, chocolates, cookies, etc. The noticeable thing here is that we always implement an inheritance relationship between classes when they have common properties. For example, in the case of products, all the products in a vending machine have names, prices, expiration dates, etc. Also, the *IS A* relationship stands valid.

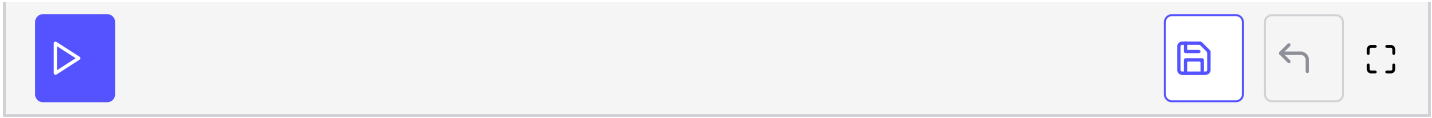
- A beverage is a product.
- Chocolate is a product.
- A cookie is a product.

So, from the above discussion, we can conclude that if there is a superclass named Product, we can derive the Beverage, Chocolate, Cookie, etc. subclasses from the

superclass.

Let's see how we would code the above classes in Dart.

```
class Product{
  String _name;
  num _price;
  String _expDate;

  Product(this._name, this._price, this._expDate);

  void printDetails(){
    print("Name: ${this._name}");
    print("Price: ${this._price}");
    print("Expiration Date: ${this._expDate}");
  }
}

class Beverage extends Product{
  num _liters;
  String _type;

  Beverage(String name, num price, String expDate, this._liters, this._type) : super(name, price,

  void beverageDetails(){
    printDetails();
    print("Liters: ${this._liters}");
    print("Type: ${this._type}");
  }
}


int main() {
  var drink = Beverage("Minute Maid", 3.50, "01/01/2020", 1.75, "Orange Juice");
  drink.beverageDetails();
}
```



Understanding the Code

The `Product` class has three instance variables, `_name`, `_price`, and `_expDate`. `Product` has a generative constructor and one method that simply prints the properties of the *product*.

On **line 15**, we are creating the subclass, `Beverage`, by *extending* the `Product` class using the keyword `extends`.

On **line 19**, we are creating a generative constructor for the `Beverage` class. To assign the first three parameters of the `Beverage` constructor, we are using the `Product` constructor. `super` is used by a subclass to refer to its superclass.

`beverageDetails()` is using the `Product` method `printDetails()`, to print the values of the properties `_name`, `_price`, and `_expDate`. It displays its own properties itself.

In the `main()` function, we are creating an instance of the `Beverage` class.

Private Instance Variables

You might have noticed that we declared all of the instance variables starting with an underscore (`_`). An underscore ensures that the instance variable is **private**. What this means is that a subclass cannot inherit that particular instance variable.

This is why we couldn't use `_name` directly in the `Beverage` class, but we were able to use `printDetails()` with ease.

Let's wrap up this chapter in the next lesson with a quiz to test what you have learned so far.