Session Goals

- User should understand the concept of classes as blueprints for creating objects. Discussing how classes encapsulate data and behavior.
- User should be able to define and use attributes within a class. Discussing visibility (public, private, protected) and how it affects access to these attributes.
- User should be able to define methods in a class, including method signatures, return types, and parameters.
- User should be able to instantiate objects from a class. Understanding the distinction between a class and an instance of that class (object).
- User should understand the role of constructors in initializing new objects.

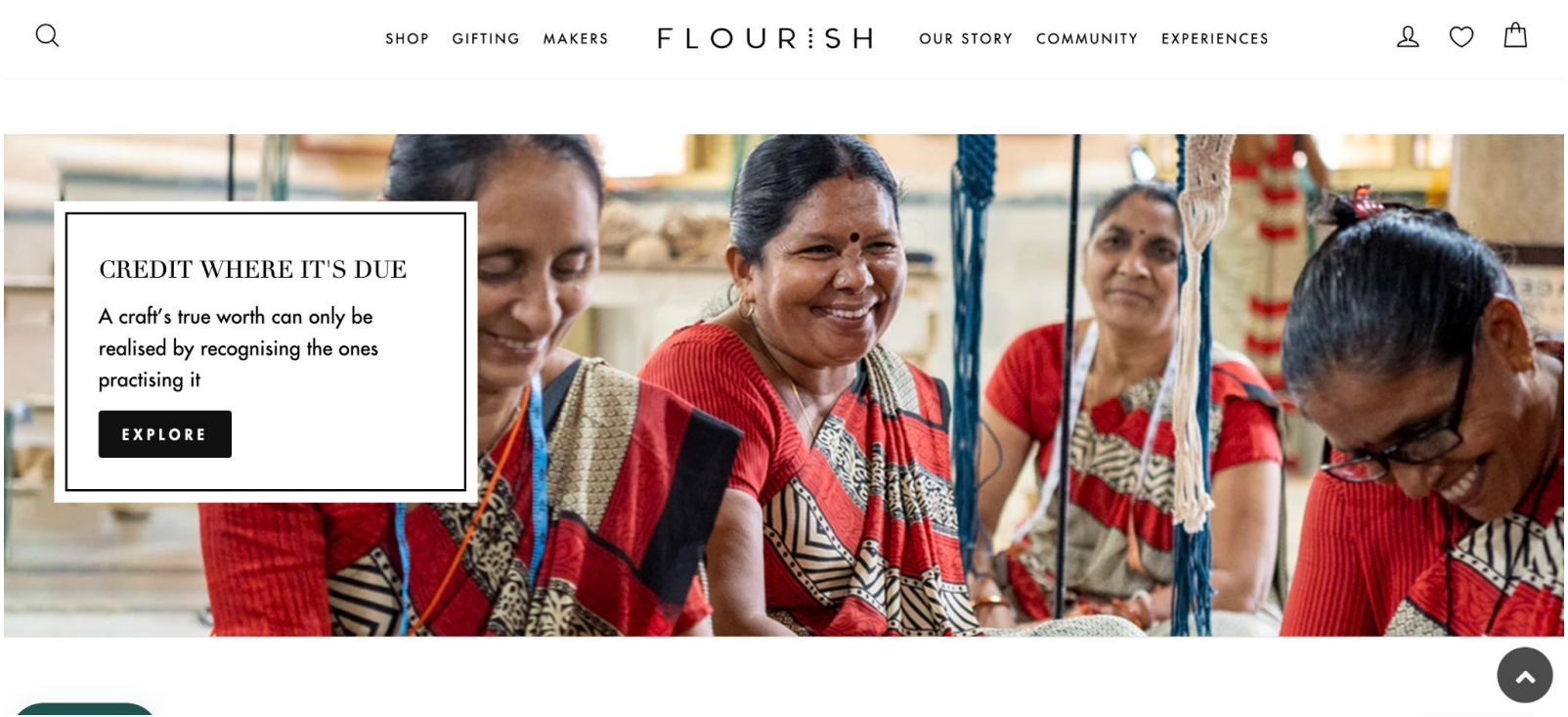
Java-111- Loops in Java

Session 7

Session Agenda

- Classes
  - Fields or attributes
  - Methods
- Class Objects
- Class Constructor
- “this” keyword

Concept #1 - Classes



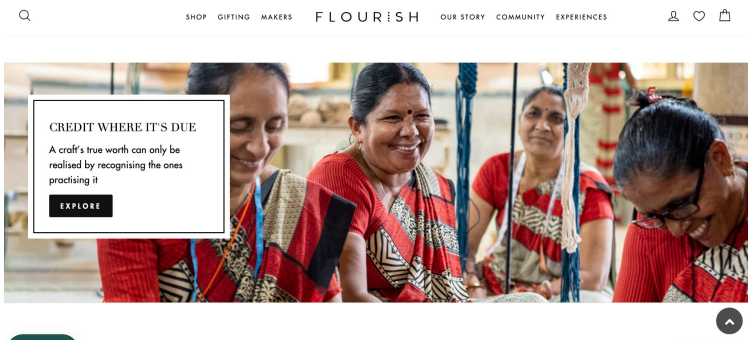
Src: https://flourish.shop/

Concept #1 - Classes, Attributes & Objects

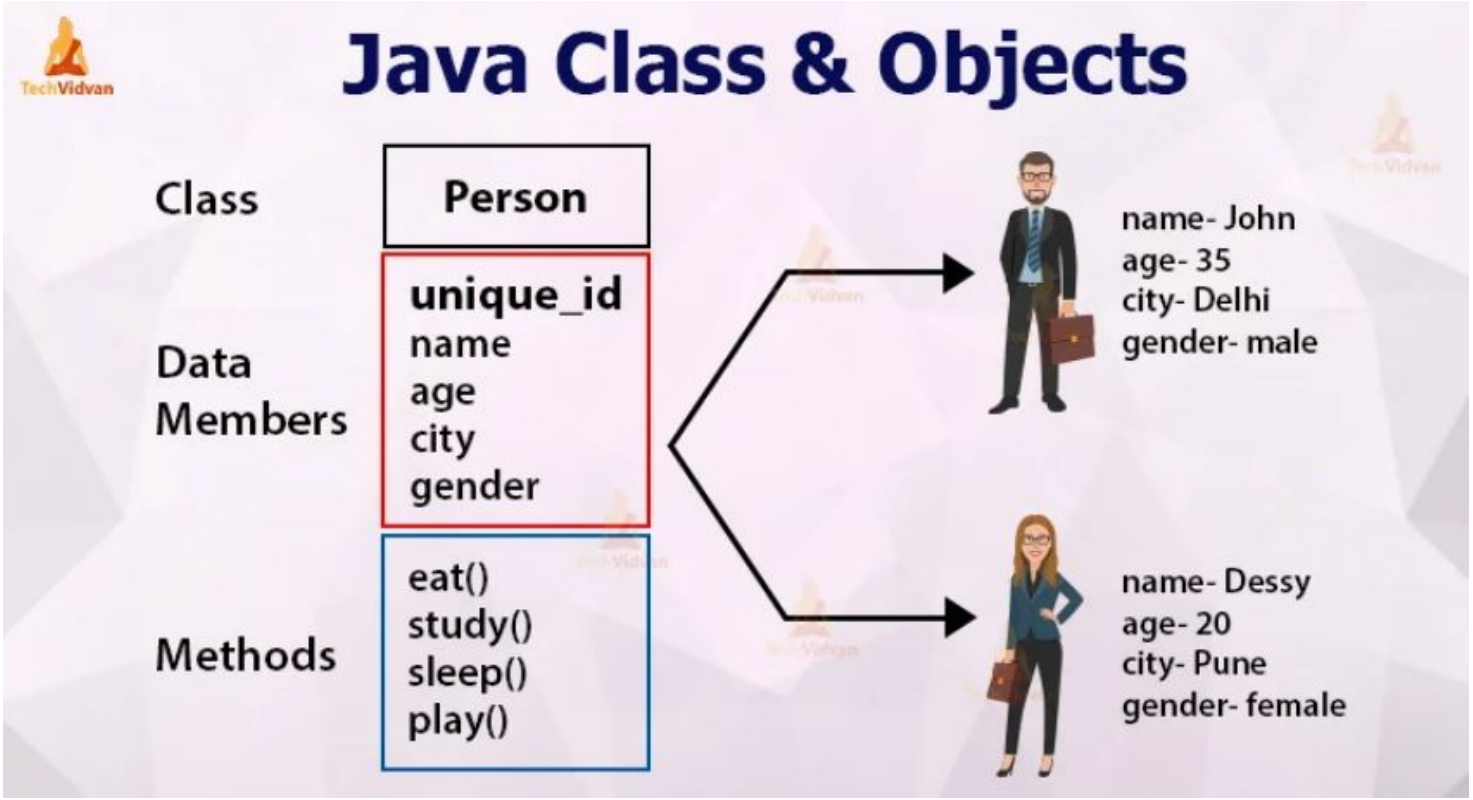
E-commerce System for **Flourish**

An e-commerce company is creating a system to manage their customers and orders. First, let's create a class for customers.

a. Add functionality to create customers for their platform.

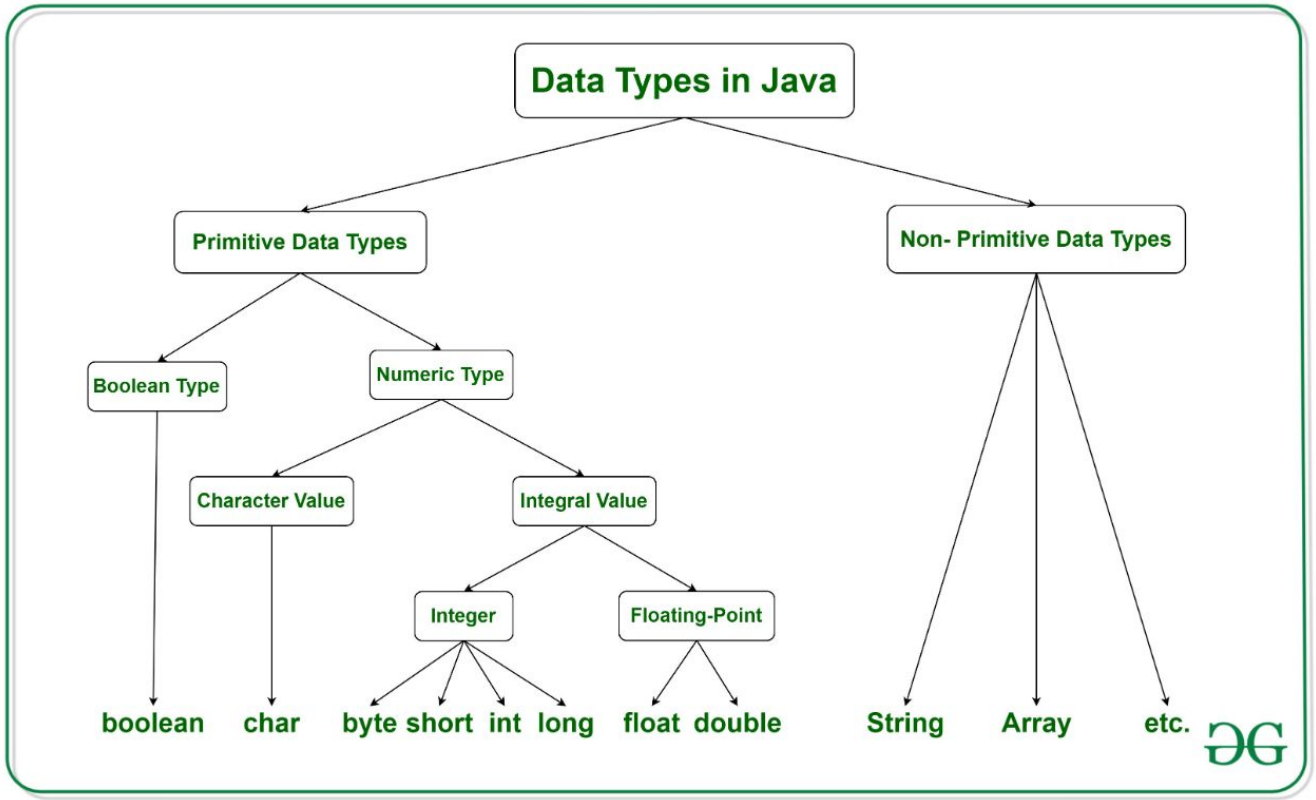


Relation between Class and Objects



Source: https://techvidvan.com/tutorials/java-class/

Concept #1 - Classes | Recap



Concept #2 - Methods

SPECIFICATIONS

PEDALS
Feimin, FB-803B, Resin Body

BRAKES
Sypo YD-V08, V-Brake, 110mm Arms, 55mm Pad Length

BRAKE LEVERS
Sypo YD-B07, Alloy, 4-Finger

HANDLEBAR
XMR, Steel, 31.8mm Dia., 0mm Rise Flat-Bar, 620mm Width

STEM
XMR, Alloy, 31.8mm Dia, 90mm Ext., 50mm Height

SADDLE
XMR Comfort Saddle, High Density Foam, Steel Rail

GRIPS
XMR, Soft PVC Grips, 130mm Length

SEAT POST
XMR, Steel, 27.2x300mm

SIZES
700c

COLOURS
Grey

Paste grind (ingredients, electricity)

Classes | Debrief

- **class** is a **User Defined Data Type**. Consists of
  - Fields (properties)
  - Methods (behaviour)
- Methods
  - Methods are functions associated with a class.
- Object is an instance of a class
  - How to create an instance? - **new()** operator
- User of the **“.”** operator to access Object Fields or Method

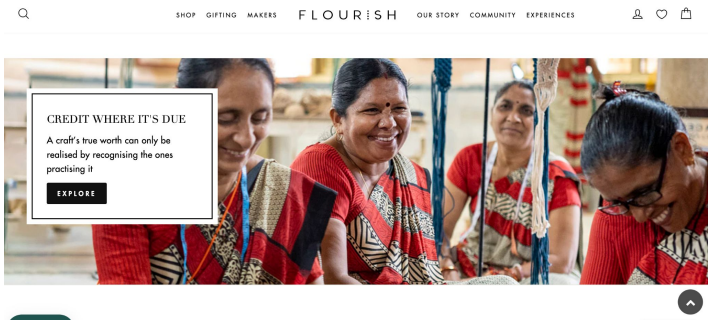
Note: We will revisit access specifiers in detail in the next Sprint. For now, we will define all fields and methods as **public**

Methods

Methods

1. Add a method to Customer class to print all the details of the customer on console.

2. Add a method to Customer class that can increase the wallet amount by a specified amount.



Parametric Methods

- Parameters are variables that *act as placeholders* for the values that are to be *input* to a method when it is called.
- When a method is defined, it is typically defined along with one or more parameters.
- The actual values that are input (or "passed") into a function when it is called are known as *arguments*.

```
class Employee{
   int id;
   String name;
   void insert(int i, String n) {
      id = i;
      name = n;
   }
   void display(){System.out.println(id+" "+name+");}
}

public class TestEmployee {
   public static void main(String[] args) {
      Employee e1=new Employee();
      e1.insert(101,"ajeet");
      e1.display();
   }
}
```

Returning values from methods

- We can use a **return** statement to send a value back from the method.

```
class Employee{
   int id;
   String name;
   void insert(int i, String n) {
      id = i;
      name = n;
   }
   String getName() {return name;}
}

public class TestEmployee {
   public static void main(String[] args) {
      Employee e1=new Employee();
      e1.insert(101,"ajeet");
      String name = e1.getName();
   }
}
```

Class Method | Debrief

- Each Method within the class has a **Method Signature**
  - Method **name**
  - **Inputs** to the Method
    - Can be empty (Non Parametric)
  - Method **return type**
    - Data Types or Class Objects
- Invoke method using **“.”** operator from the class object
- Pass the necessary inputs based on the method signature
- Consume the return value if not **void**

5 minute break

Concept #3 - Constructor

This is a special method in a class

Can optionally take input parameters

Set and Get methods

- The **get** method **returns the variable value**, and the **set** method **sets the value**.

```
public class Person {
  private String name; // private = restricted access

  public String getName() {
    return name;
  }
  public void setName(String newName) {
    this.name = newName;
  }
}

public class Main {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.name = "John";  // error
    System.out.println(myObj.name); // error

    myObj.setName("John"); // Use Setter
    System.out.println(myObj.getName()); // User Getter
  }
}
```

Activity: Implement GetBeets() and SetBeets()

Link

Concept #4 - this

- **this** is a reference variable that refers to the current object, in the constructor or other methods
- Why do we need this?
  - How would you differentiate between a class member variable and a variable within a method, with the same name as the class member variable?

```
public class Main {
  int x;

  // Constructor with a parameter
  public Main(int x) {
    this.x = x;
  }

  // Call the constructor
  public static void main(String[] args) {
    Main myObj = new Main(5);
    System.out.println("Value of x = " + myObj.x);
  }
}
```

What output would you see, if you remove **this** and run the above code?

Concept - this

Which of the following is/are true about 'this' (in the context of class)?

1. When we use ‘this’ in a class, it refers to that particular instance/class object
2. 'this' can be used to refer to both fields and methods of that class

Activity: Return the total listener count

Do it from scratch

Link

Ignore these keywords for now

public

private

static

These will be take up in the next sprint when we get to the OOP concepts.
For now **just create all attributes and methods as public**.

Further Reading

- Object and Class in Java
- Method in Java
- Constructor
- this

Note that some of the aspects in the references above, will be covered in the next sprint since they are slightly advanced topics.

Keep
Learning,
Keep
Coding.