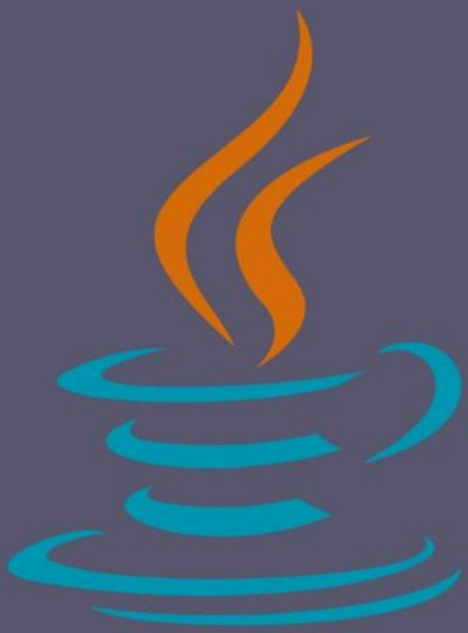
OBJECT ORIENTED PROGRAMMING USING



Java

Let's explore technology
together to live in the future

Checkout more on
https://github.com/Sy-hash-collab

Sy-hash-collab

Java Abstraction:

The major use of abstract classes and methods is to acheive abstraction in Java. Abstraction allows us to hide the unnecessary details and only show needed information.
Abstraction can be acheived by: abstract classes or interfaces.

- Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- Abstract Method: can only be used in abstract class and it doesn't have a body. The body is provided by subclass.

An abstract class is a class that's declare with abstract keyword.
An abstract method is a method that is declared without implementation.
Abstract class may or maynot have all abstract methods, some of them can be concrete methods.

* A method defined abstract must always be redefined in the subclass, thus making overriding compulsory or making the subclass itself abstract.

An abstract class can have parameterized constructors and default constructor is always present in abstract class

Java Abstract Class:

The abstract class in Java cannot be instantiated (we cannot create objects of abstract classes).

=> We use 'abstract' keyword to declare abstract class.

```
// create an abstract class
abstract class Language {

}
// try to create object // gives error
Language obj = new Language();
```

=> An abstract class can have both regular methods and abstract methods.

```
abstract class Language
{   //abstract method
    abstract void method1();
    // regular method
    void method2()
    {
        System.out.println("This is a
                            regular method");
    }
}
```

Java Abstract Method:

A method that doesn't have its body is known as abstract method.

- We use the same abstract keyword to create abstract methods.

```
abstract void display();
```

Here display() is abstract method. The body of display() is replaced by ';'.

=> If a class contains an abstract method, then the class should be declared abstract Otherwise it gives error.

```
class Language //error // class should be abstract
{
  abstract void method1();
}
```

=> Though abstract classes can't be instantiated we can create subclasses from it. We can access members of abstract class using object of subclass

```
abstract class Language
{  //method of abstract class.
   public void display()
   {
     System.out.println("Java");
   }
}
class Main extends Language {
  public static void main(String[]args)
  //create object of main class
  {  Main m = new Main();
     m.display(); //access method of abstract class using obj. of main class
  }
```

Implementing Abstract Methods:

If the abstract class includes any abstract method, then all the child classes inherited from the abstract superclass must provide the implementation of abstract method.

```
abstract class Animal
 {
    abstract void makeSound();

    public void eat()
     { System.out.println("I can eat");
     }
}
pu class Dog extends Animal
   {
      public void makeSound()
     { System.out.println("Bark, bark");
      }
   }
public class Main {
 public static void main(String[]args)
  {
    Dog d = new Dog();

  d.eat();      d.makeSound();
  d.display();
}}
```

Interfaces in Java:

Interface in Java is a mechanism to achieve abstraction

There can be only abstract method in Java interface, not the method body.

An interface is a completely abstract class that is used to group related methods with empty bodies.

By interface, we can support functionality of multiple inheritance.

To access interface methods, interface must be "implemented" by another class with implements keyword. The body of interface method is provided by implement class.

:> Like abstract classes, interfaces can't be used to creat objects.

Interface methods don't have a body. The body is provided by implement class. On implementation, it must override all methods of interface.

Interface methods are by-default abstract public

```
interface Addmission
{
public abstract void getData();
public abstract void display();
}
```

Interface attributes are by default public, static and final.

```
interface Admission
{ public static final int x;

}
```

* a class can extends and implements at the same time.

```
interface Admission
{ void getData();
  void getDisplay();
}
interface Exam
{ void issueRollnoSlips();
}
class MidTermExam implements Admission, Exam
{ void getData()
  { }
  void getDisplay()
  { }
  Void issueRollnoSlips()
  { }
}
class Main {
public static void main(String[] args)
{
MidTermExam e = new MidTermExam();
  e.getData();
  e.getDisplay();
  e.issueRollnoSlips();
```