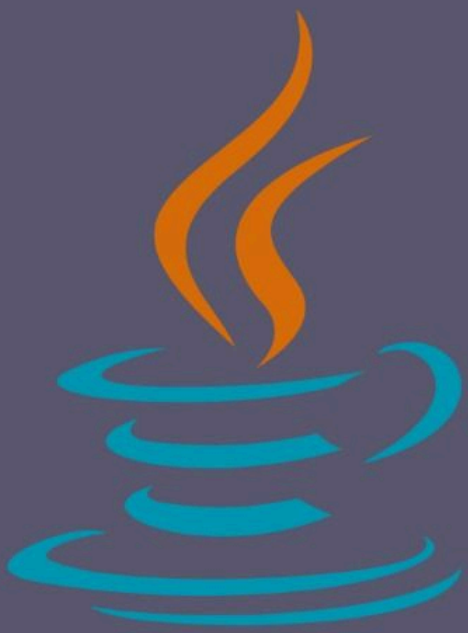
OBJECT ORIENTED PROGRAMMING USING



Java

Let's explore technology
together to live in the future

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

Sy-hash-collab

10/10/23

Polymorphism: Polymorphism means "many forms".

It occurs when we have many classes that are related to eachother by inheritance. Inheritance lets us inherit attributes and methods from another class, Polymorphism uses those methods to perform different tasks. Polymorphism allows you to define one interface and have multiple implementations.

Definition:

"When same entity (method, operator, object) can perform different operations in different situations."

\+ → As a Method takes parameters
\+ → As an operator takes operands.

```
class Animal
{ public void animalSound()
  { System.out.println("Animal makes
                        sound");
  }
}

class Cat extends Animal
{ public void animalSound()
  { System.out.println("Meow, meow");
  }
}

class Dog extends Animal
{ public void animalSound()
    System.out.println("Bow, bow");
}}
```

- Compile-Time Polymorphism
- Method overloading → Different Signature

```
class Pattern
{   // method with no parameters
    public void display()
    {  for (int i=0 ; i<10 ; i++)
       { System.out.println("*"); }
    }
    // method with one parameter
    public void display(char symbol)
    {  for(int i=0 ; i<10 ; i++)
       { System.out.println(symbol); }
    }
}

class Main {
public static void main(String[] args)
    {  Pattern p = new Pattern();

    p.display(); // calling method having no parameters
     System.out.println("\n");

    p.display('#');

    }
}
```

Output:

```
* * * * * * * * * *
# # # # # # # # # #
```

Key Points:

=> Method overloading is done in same class only.

=> Can't be used with inheritance as inheritance involve parent, child classes

• Run-Time Polymorphism:

Method Overriding: → Same Signature

When a child class "implement a metho
from parent class

```
class Language
{
  public void displayInfo()
    { System.out.println ("Common English
                          Language");
  }
}

class Java extends Language
{  "overriding super-class" method.
  public void displayInfo()
    { System.out.println ("Java programmi
                          language");
  }
}

class Main{
public static void main (String [] args
{   "create an object of Java class
    Java j = new Java();

     j.displayInfo();
    " create an object of Language class
    Language l = new Language();

     l.displayInfo();
}
}
```

How Polymorphic method is made?

```
class Animal
{
  public void animalSound ()
  { System.out.println ("Animal makes
                         sound");
  }}
class Cow extends Animal
{
  public void animalSound()
  { System.out.println (" Baa, baa ");}
}
class Dog extends Animal
{
  public void animalSound()
  { System.out.println ("Bow, bow");}
}
class Main {
public static void main (String [] args)
{
  Animal a = new Animal();
  Cow c = new Cow();
  Dog d = new Dog()
  a.animalSound();
  c.animalSound();
  d.animalSound();
```

=> animalSound() is a polymorphic method and it behaves differently because of:
- Different Calling Objects (Animal, Cow, Dog)
- Different Contexts
- Different Situations

In inheritance, Parent Class can be used as reference to make Object of child class

```
class Main {
public static void main (String [] args)
{
    Animal a1 = new Animal ();
    Animal a2 = new Cow ();
    Animal a3 = new Dog ();

    a1.animalSound ();
    a2.animalSound ();
    a3.animalSound ();
}
}
```

```
Animal a3 = new Dog();
```
(reference of parent class) (object of child class)

Note: If they're not parent-child classes (no inheritance) then you cann't use it as this will give error.

=> In inheritance, the child class has two methods, one overrided method other is inherited from parent class.

Cow, Dog has two animalSound() methods i-e overrided method and other is inherited from Animal

'Super' keyword:

If you want to call both overidded and inherited methods of child class then you should use "super" keyword such as:

```
class Animal
{
    public void animalSound()
    { System.out.println("Animal makes
                                sound");
    } }
class Dog extends Animal
{   public void animalSound()
    { System.out.println("Bow, bow");
      super.animalSound();
    }
}
```

• Operator Overloading:

+ → numeric addition
+ → string concatenation

&, |, ! → logical and bitwise operations

1) When '+' is used with numbers integers, floating point numbers it performs mathematical addition.

```
int a = 5;
int b = 6;
int sum = a+b; // 11
```

When '+' is used with "strings", it will perform concatenation.

```
String first = "Java";
String second = "Programming";

name = first + second; // JavaProgramming
```

• Polymorphic Variables:

A variable is called polymorphic if it refers to different values under different conditions.

Object variables (instance variables) represent the behaviour of polymorphic variables in Java. It is because object variables of a class can refer to objects of its class as well as objects of its subclasses.

```
class ProgrammingLanguage
{
  public void display()
  { System.out.println("I am
                  Programming Language");
  }
}
class Java extends ProgrammingLanguage
{  @Override
   public void display()
   { System.out.println("I am Object
                  Oriented Programming
                        Language");
}}

class Main
{ public static void main(String[]args)
   {  //declare an object variable
      ProgrammingLanguage p1;

//create object of Programminglanguage
      p1 = new ProgrammingLanguage();
       p1.display();
//  create object of Java
      p1 = new ProgrammingLanguage();
       p1.display();

}}
```

Compile Time Polymorphism (Static Polymorphism)

=> Achieved through "method overloading", where multiple methods have same name but different parameters.

- The "compiler" determines which method to invoke based on the number and types of arguments passed during "method call"
- It is also known as early, static binding.

```
class Calculator {
  public int add(int a, int b)
  { return a+b; }

  public doble add (double a, double b)
  { return a+b; }
}
```

Run-Time Polymorphism (Dynamic Polymorphism)

=> Achieved through "method overriding", where a sub-class provides a specific implementation of a method, that's already defined in its superclass.

- The "JVM" determines which method to invoke based on the "actual type of object being referred to", "at runtime"
- It is also known as late or dynamic binding

```
class Animal {
  public void sound() {
  } System.out.println("Animal makes sound
}

  class Dog extends Animal
  public void sound() { System.out.println("Bou
```