

# Inheritance

## Fundamentals to programming I

Glenny Shínderly Choque Vilcape  
Edisson Franklin Checalla Soto

<sup>1</sup>System Engineering School  
System Engineering and Informatic Department  
Production and Services Faculty  
San Agustin National University of Arequipa

2020-08-04



# Content

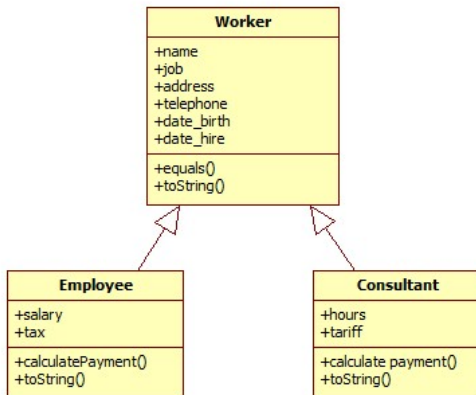
- 1 Introduction
- 2 Basic Example 1
- 3 Method override
- 4 Basic Example 2
- 5 Advanced Example 1
- 6 Advanced Example 2
- 7 References

# Introduction

- The mechanism known as inheritance allows reusing classes: A new class is created that extends the functionality of an existing class without having to rewrite the code associated with it.

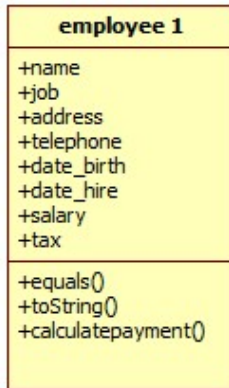
# Inheritance uml diagram

the employee and consultant classes, in addition to the attributes and operations they define, inherit all their attributes and operations from the worker.



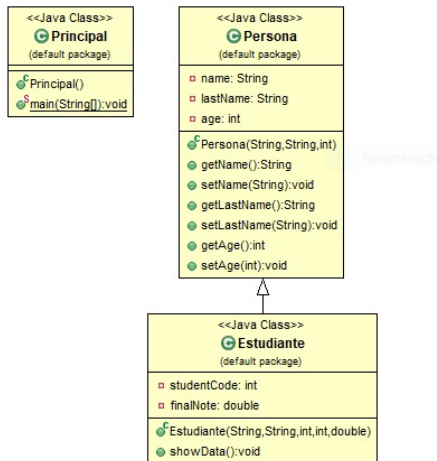
## Employee uml diagram

A particular employee will have, in addition to his or her attributes and operations as an employee, all the attributes corresponding to the worker superclass.



# UML Basic Example 1

- <https://youtu.be/4gdhl-iRZZg>



# Basic Examples 1- class Person (Part 1)

## Listing 1: Persona.java

```
public class Persona {  
    private String name;  
    private String lastName;  
    private int age;  
  
    public Persona(String name, String lastName, int age) {  
        this.name = name;  
        this.lastName = lastName;  
        this.age = age;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

## Basic Examples 1 - class Person (Part 2)

### Listing 2: Persona.java

```
public String getLastName() {  
    return lastName;  
}  
  
public void setLastName(String lastName) {  
    this.lastName = lastName;  
}  
  
public int getAge() {  
    return age;  
}  
  
public void setAge(int age) {  
    this.age = age;  
}  
}
```



# Basic Examples 1 - class Student

## Listing 3: Estudiante.java

```
public class Estudiante extends Persona{
    private int studentCode;
    private double finalNote;
    public Estudiante (String name, String lastName, int
        age, int studentCode, double finalNote) {
        super(name,lastName,age);
        this.studentCode=studentCode;
        this.finalNote=finalNote;
    }
    public void showData() {
        System.out.println("Name: " + getName() + "\nLastname
            : " + getLastName() +
            "\nAge: " + getAge() + "\nStudent Code: " +
            studentCode + "\nFinal Note: " + finalNote);
    }
}
```

# Basic Examples 1 - class Principal

## Listing 4: Principal.java

```
public class Principal {  
    public static void main(String []args) {  
        Estudiante student = new Estudiante("Glenny", "Choque",  
            18, 123, 18.3);  
        student.showData();  
    }  
}
```

## BASic Example 1- Program Run

```
1 |
2 public class Principal {
3 public static void main(String []args) {
4     Estudiante student = new Estudiante("Glenny", "Choque", 18, 123, 18.3);
5     student.showData();
6 }
7 }
8
```

Console ✕

<terminated> Principal [Java Application] C:\Program Files\Java\jre1.8.0\_251\bin\javaw.exe (3 ago. 2020 15:2

Name: Glenny

Lastname: Choque

Age: 18

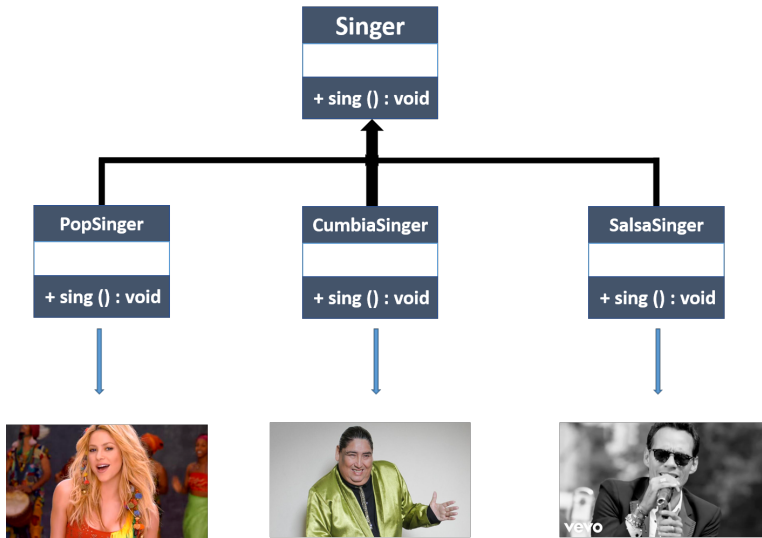
Student Code: 123

Final Note: 18.3

# Method override

- A subclass inherits all the methods of its superclass that are accessible to that subclass unless the subclass overwrites the methods.
- A subclass overwrites a method of its superclass when it defines a method with the same characteristics (name, number, and type of arguments) as the method of the superclass.
- Subclasses use method overwriting most of the time to add or modify the functionality of the parent class's inherited method.

## Basic Example 2 - UML Diagram Representation



## Basic Examples 2- class Singer

### Listing 5: Singer.java

```
public class Singer {  
  
    public void sing () {  
        System.out.println("i'm singing");  
    }  
}
```

## Basic Examples 2- class PopSinger

### Listing 6: PopSinger.java

```
public class PopSinger extends Singer {  
  
    @Override  
    public void sing () {  
        System.out.println("i'm singing pop");  
    }  
}
```

## Basic Examples 2- class SalsaSinger

### Listing 7: SalsaSinger.java

```
public class SalsaSinger extends Singer {  
    @Override  
    public void sing () {  
        System.out.println("i'm singing salsa");  
    }  
}
```



## Basic Examples 2- class CumbiaSinger

### Listing 8: CumbiaSinger.java

```
public class CumbiaSinger extends Singer {  
    @Override  
    public void sing () {  
        System.out.println("i'm singing cumbia");  
    }  
}
```

## Basic Examples 2- class CumbiaSinger

### Listing 9: Main.java

```
public class Main {  
  
    public static void main(String[] args) {  
        Singer singer = new Singer ();  
        PopSinger popSinger = new PopSinger ();  
        SalsaSinger salsaSinger = new SalsaSinger ();  
        CumbiaSinger cumbiaSinger = new CumbiaSinger ();  
  
        singer.sing();  
        popSinger.sing();  
        salsaSinger.sing();  
        cumbiaSinger.sing();  
    }  
}
```

## Basic Example 2 - Program Run

```
2 public class Main {  
3  
4     public static void main(String[] args) {  
5         Singer singer = new Singer ();  
6         PopSinger popSinger = new PopSinger ();  
7         SalsaSinger salsaSinger = new SalsaSinger ();  
8         CumbiaSinger cumbiaSinger = new CumbiaSinger ();  
9  
10        singer.sing();  
11        popSinger.sing();  
12        salsaSinger.sing();  
13        cumbiaSinger.sing();  
14    }  
15 }  
16
```



Problems



Javadoc



Declaration



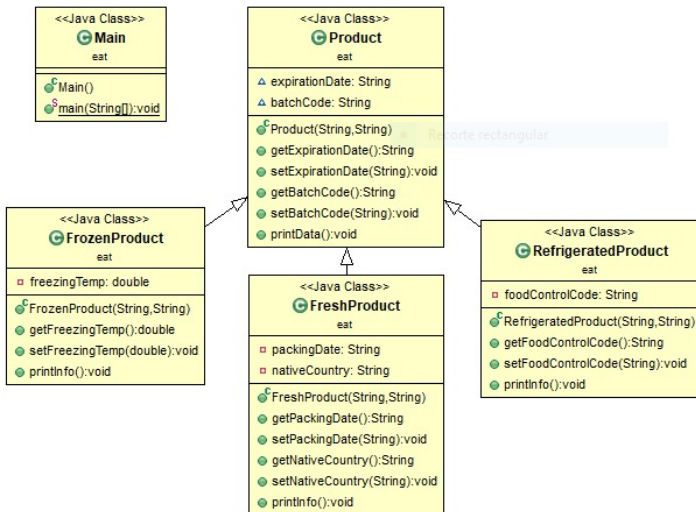
Console



```
<terminated> Main (8) [Java Application] C:\Program Files\Java\jre1.8.0_261\bin\javaw.exe  
i'm singing  
i'm singing pop  
i'm singing salsa  
i'm singing cumbia
```

# Advanced Example 1 - UML Diagram Representation

- <https://Pordefirnolohiceporvago>



# Advance Example 1- class product (Part 1)

## Listing 10: Product.java

```
package eat;

public class Product {
    String expirationDate, batchCode;

    public Product(String expirationDate, String batchCode)
    {
        this.expirationDate = expirationDate;
        this.batchCode = batchCode;
    }

    public String getExpirationDate() {
        return expirationDate;
    }

    public void setExpirationDate(String expirationDate) {
```

## Advance Example 1- class product (Part 2)

### Listing 11: Product.java

```
        this.expirationDate = expirationDate;
    }
    public String getBatchCode() {
        return batchCode;
    }
    public void setBatchCode(String batchCode) {
        this.batchCode = batchCode;
    }
    public void printData() {
        System.out.println("Expiration Date: " +
            getExpirationDate() + "\nBatch Code: " +
            getBatchCode());
    }
}
```

## Advance Example 1- class FrozenProduct

Listing 12: FrozenProduct.java

```
package eat;

public class FrozenProduct extends Product{
    private double freezingTemp;
    public FrozenProduct(String expirationDate, String
        batchCode){
        super(expirationDate, batchCode);
        freezingTemp = 0;
    }
    public double getFreezingTemp() {
        return freezingTemp;
    }
    public void setFreezingTemp(double Temp) {
        freezingTemp = Temp;
    }
    public void printInfo(){
        printData();
        System.out.println("Freezing temperature: " +
            getFreezingTemp() + "\n");
    }
}
```

# Advance Example 1- class FreshProduct (Part 1)

## Listing 13: FreshProduct.java

```
package eat;

public class FreshProduct extends Product {
    private String packingDate, nativeCountry;
    public FreshProduct (String expirationDate, String
        batchCode) {
        super(expirationDate, batchCode);
        packingDate = "Desconocido";
        nativeCountry = "Desconocido";
    }
    public String getPackingDate() {
        return packingDate;
    }
    public void setPackingDate(String Date) {
```



## Advance Example 1- class FreshProduct (Part 2)

### Listing 14: FreshProduct.java

```
        packingDate = Date;
    }
    public String getNativeCountry() {
        return nativeCountry;
    }
    public void setNativeCountry(String Country) {
        nativeCountry = Country;
    }
    public void printInfo() {
        printData();
        System.out.println("packing Date: " +
            getPackingDate() + "\nNative Country" +
            getNativeCountry()+"\n");
    }
}
```

## Advance Example 1- class RefrigeratedProduct

Listing 15: RefrigeratedProduct.java

```
package eat;

public class RefrigeratedProduct extends Product{
    private String foodControlCode;
    public RefrigeratedProduct(String expirationDate,
        String batchCode){
        super(expirationDate, batchCode);
        foodControlCode = "Desconocido";    }
    public String getFoodControlCode() {
        return foodControlCode;            }
    public void setFoodControlCode(String Code) {
        foodControlCode = Code;            }
    public void printInfo(){
        printData();
        System.out.println("food Control Code: " +
            getFoodControlCode() + "\n" );
    }
}
```

## Advance Example 1- class Main (Part 1)

### Listing 16: Main.java

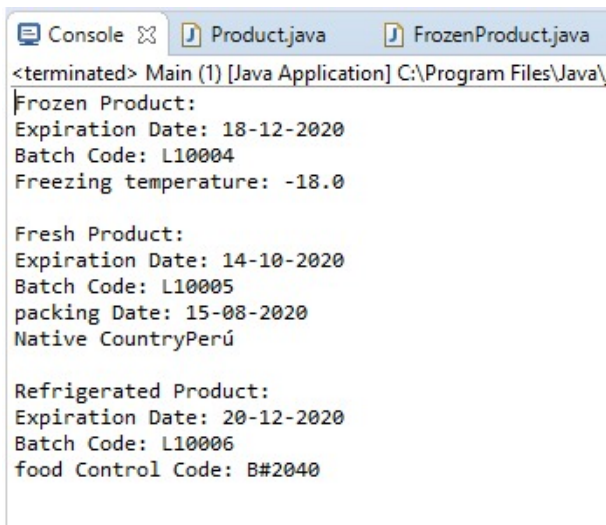
```
public class Main {  
    public static void main (String[]args) {  
        FrozenProduct nuggets = new FrozenProduct ("18-12-2020"  
            , "L10004");  
        FreshProduct lettuce = new FreshProduct ("14-10-2020",  
            "L10005");  
        RefrigeratedProduct juice = new RefrigeratedProduct ("20-12-2020", "L10006");  
        nuggets.setFreezingTemp(-18);  
        lettuce.setPackingDate("15-08-2020");  
        lettuce.setNativeCountry("Peru");  
        juice.setFoodControlCode("B#2040");  
        System.out.println("Frozen Product: ");  
        nuggets.printInfo();label
```

## Advance Example 1- class Main (Part 2)

### Listing 17: Main.java

```
System.out.println("Fresh Product: ");  
lettuce.printInfo();  
System.out.println("Refrigerated Product: ");  
juice.printInfo();  
}  
}
```

## Advance Example 1- Program Run



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\
Frozen Product:
Expiration Date: 18-12-2020
Batch Code: L10004
Freezing temperature: -18.0

Fresh Product:
Expiration Date: 14-10-2020
Batch Code: L10005
packing Date: 15-08-2020
Native CountryPerú

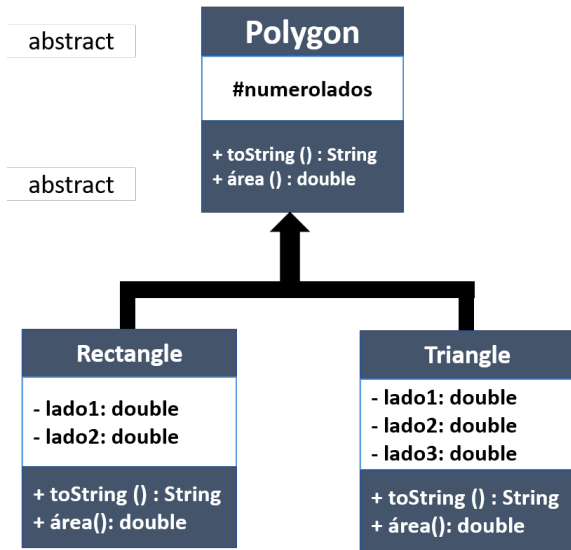
Refrigerated Product:
Expiration Date: 20-12-2020
Batch Code: L10006
food Control Code: B#2040
```

## Advanced Example 2 -statement

Make a program to calculate the area of Polygons (Triangles and Rectangles) the program should be able to store in an N array triangles and rectangles, and at the end show the area and data of each one. To do this, you will have the following:

- A super class called Polygon.
- A sub class called Rectangle.
- A sub class called Triangle.

## Advanced Example 2 - UML Diagram Representation



## Advance Example 2- class Polygon (Part 1)

### Listing 18: Polygon.java

```
public abstract class Polygon {  
    protected int numSides;  
  
    public Polygon(int numsides) {  
        this.numSides = numsides;  
    }  
    public int getNumsides() {  
        return numSides;  
    }  
  
    public void setNumsides(int numsides) {  
        this.numSides = numsides;  
    }  
}
```



## Advance Example 2- class Polygon (part 2)

### Listing 19: Polygon.java

```
}  
  
public String toString() {  
    return "Number Of sides: " + numSides ;  
}  
//declaramos el metodo area como abstracto  
public abstract double area();  
}
```

## Advance Example 2- class Rectangle (Part 1)

### Listing 20: Rectangle.java

```
public class Rectangle extends Polygon {  
  
    private double side1;  
    private double side2;  
    public Rectangle(double side1, double side2) {  
        super(2);  
        this.side1 = side1;  
        this.side2 = side2;  
    }  
    public double getSide1() {  
        return side1;  
    }  
    public void setSide1(double side1) {  
        this.side1 = side1;  
    }  
}
```

## Advance Example 2- class Rectangle (Part 2)

### Listing 21: Rectangle.java

```
public double getSide2() {  
    return side2;  
}  
  
public void setSide2(double side2) {  
    this.side2 = side2;  
}  
  
@Override  
public String toString() {  
    return "Rectangle: \n" + super.toString() + "\nside1=  
        " + side1 + "\nside2= " + side2;  
}  
  
@Override  
public double area() {  
    return side1 * side2;  
}
```

## Advance Example 2- class Triangle(Part 1)

### Listing 22: Triangle.java

```
public class Triangle extends Polygon{
    private double side1;
    private double side2;
    private double side3;

    public Triangle( double side1, double side2, double
        side3) {
        super(3);
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }

    public double getSide1() {
        return side1;
    }
}
```

## Advance Example 2- class Triangle(Part 2)

### Listing 23: Triangle.java

```
public void setSide1(double side1) {  
    this.side1 = side1;  
}  
  
public double getSide2() {  
    return side2;  
}  
  
public void setSide2(double side2) {  
    this.side2 = side2;  
}  
  
public double getSide3() {  
    return side3;  
}  
}label
```

## Advance Example 2- class Triangle(Part 3)

### Listing 24: Triangle.java

```
public void setSide3(double side3) {  
    this.side3 = side3;  
}  
  
@Override  
public String toString() {  
    return "Triangle: \n" + super.toString() + " \nside1=  
    " + side1 + "\nside2=" + side2 + "\nside3=" +  
    side3;  
}  
  
@Override
```

## Advance Example 2- class Triangle(Part 4)

Listing 25: Triangle.java

```
public double area() {  
    //formula de Heron  
    double p = (side1+side2+side3)/2;  
    return Math.sqrt (p*(p-side1)*(p-side2)*(p-side3));  
}  
}
```

## Advance Example 2- class Main(Part 1)

### Listing 26: Main.java

```
import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    static Scanner sc = new Scanner (System.in);
    static ArrayList<Polygon> polygon = new ArrayList <
        Polygon> ();
```



## Advance Example 2- class Main(Part 2)

### Listing 27: Main.java

```
public static void main(String[] args) {  
  
    //llenarPoligono  
    fillPolygon();  
    //mostrar datos del area de cada poligono  
    showResults();  
}
```

## Advance Example 2- class Main(Part 3)

### Listing 28: Main.java

```
public static void fillPolygon() {  
    char answer;  
    int option;  
    do {  
        do{  
            System.out.println("Type in your desired polygon"  
                                );  
            System.out.println("1. Triangle");  
            System.out.println("2. Rectangle");  
            System.out.print("option: ");  
            option = sc.nextInt();  
        }while (option<1 || option>2);  
    }
```

## Advance Example 2- class Main(Part 4)

### Listing 29: Main.java

```
switch (option) {  
    case 1: fillTriangle(); //llenar un triangulo  
        break;  
    case 2: fillRectangle(); //llenar un rectangulo  
        break;  
}  
System.out.println("\nDo you want to introduce  
    another polygon (y/n)?");  
    answer = sc.next().charAt(0);  
    System.out.println("");  
} while (answer=='y' || answer=='Y');
```

## Advance Example 2- class Main(Part 5)

### Listing 30: Main.java

```
}

public static void fillTriangle() {
    //solicitar lados
    double side1, side2, side3;
    System.out.print("\nenter the side 1 of the triangle:
        ");
    side1= sc.nextInt();
    System.out.print("\nenter the side 2 of the triangle:
        ");
    side2= sc.nextInt();
    System.out.print("\nenter the side 2 of the triangle:
        ");
    side3= sc.nextInt();
}
```

## Advance Example 2- class Main(Part 6)

### Listing 31: Main.java

```
Triangle triangle = new Triangle (side1,side2, side3)
;
//guardamos un triangulo dentro de nuestro arreglo de
    poligonos
polygon.add(triangle);
}

public static void fillRectangle() {
    double side1, side2;
    System.out.print("\nenter the side 1 of the Rectangle
        : ");
    side1= sc.nextInt();
    System.out.print("\nenter the side 2 of the Rectangle
        : ");
```

## Advance Example 2- class Main(Part 7)

### Listing 32: Main.java

```
side2= sc.nextInt();

Rectangle rectangle = new Rectangle (side1,  side2);
//guardamos un rectangulo dentro de nuestro arreglo
    de poligonos
polygon.add(rectangle);
}
public static void  showResults() {
    //recorriendo el arreglo de poligonos
    for (Polygon poly: polygon ) {
        System.out.println(poly.toString());
        System.out.println(("The area is: " + poly.area()))
        ;
    }
}
```

## Advance Example 2- class Main(Part 7)

### Listing 33: Main.java

```
        System.out.println("");  
    }  
}  
}
```

## Advance Example 2- Program Run

```
Type in your desired polygon
1. Triangle
2. Rectangle
option: 2

enter the side 1 of the Rectangle: 5
enter the side 2 of the Rectangle: 7

Do you want to introduce another polygon (y/n)?
y

Type in your desired polygon
1. Triangle
2. Rectangle
option: 1

enter the side 1 of the triangle: 5
enter the side 2 of the triangle: 6
enter the side 2 of the triangle: 7

Do you want to introduce another polygon (y/n)?
n

Rectangle:
Number Of sides: 2
side1= 5.0
side2= 7.0
The area is: 35.0

Triangle:
Number Of sides: 3
side1=5.0
side2=6.0
side3=7.0
The area is: 14.696938456699069
```



## References - Web pages

- <https://elvex.ugr.es/decsai/java/pdf/9B-herencia.pdf>
- <https://elvex.ugr.es/decsai/java/>
- <https://aprenderaprogramar.com/foros/index.php?topic=2376.0>
- <https://rua.ua.es/dspace/bitstream/10045/15995/1/POO-3-Herencia-10-11.pdf>



Thanks!...