Inheritance

Fundamentals to programming I

Glenny Shínderly Choque Vilcape
Edisson Franklin Checalla Soto

[1]System Engineering School
System Engineering and Informatic Department
Production and Services Faculty
San Agustin National University of Arequipa

2020-08-04

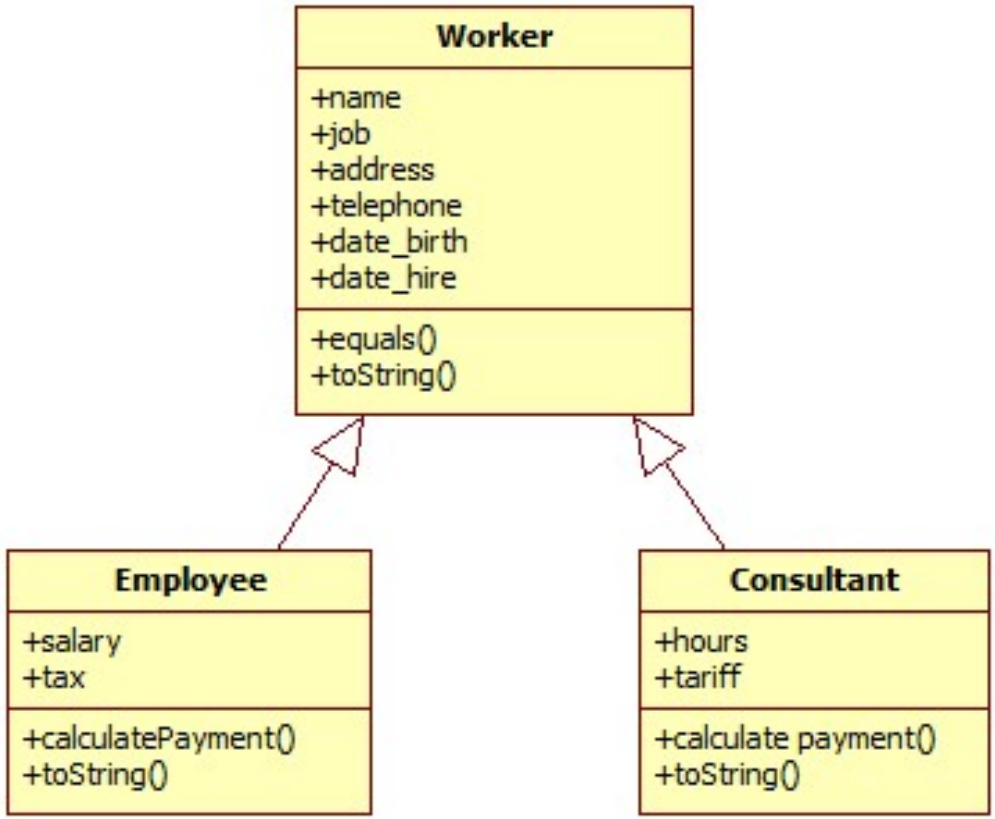
Content

Introducction

- 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.

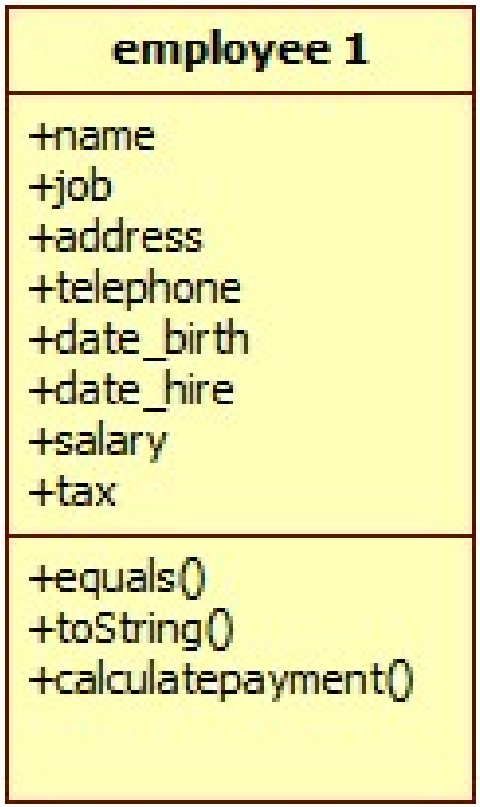
Inheritance uml diagram

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



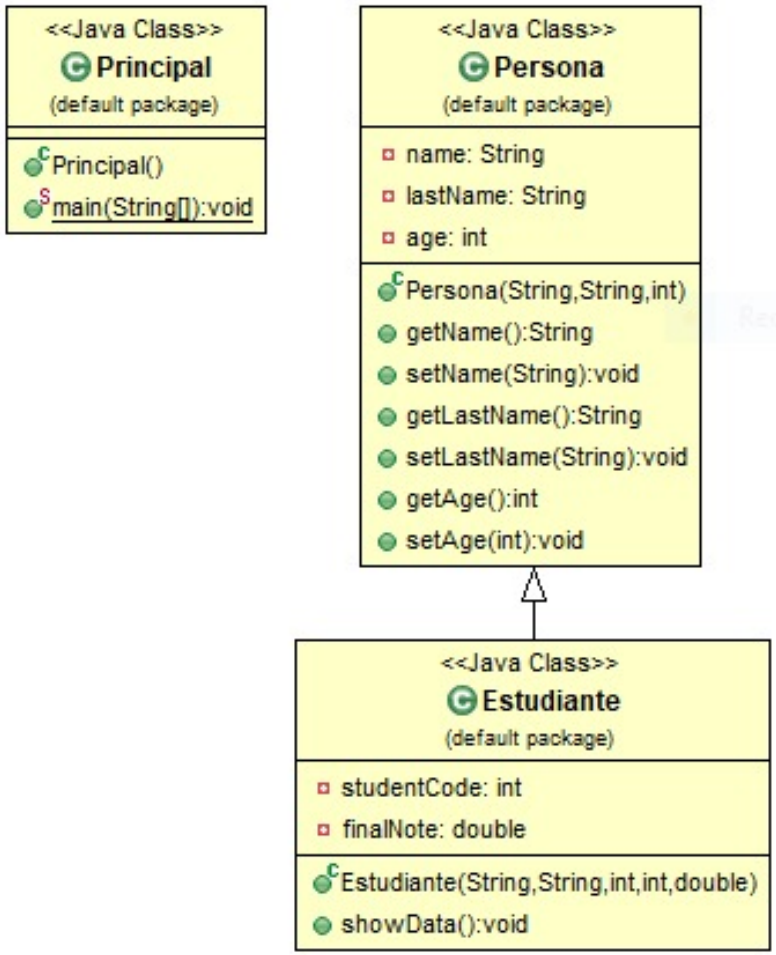
Employee uml diagram

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();
}
}
```

References - Web pages

- https://elvex.ugr.es/decsai/java/
- https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html
- https://www.eclipse.org/downloads/packages/release/2020-06/r/eclipse-ide-enterprise-java-developers
- https://www.objectaid.com/home
- https://openjdk.java.net/install/index.html
- https://code.visualstudio.com/
- https://www.sublimetext.com/
- https://vimhelp.org/
- https://www.computerscience.gcse.guru/theory/von-neumann-architecture
- https://stackoverflow.com/questions/48304498/are-wrappers-of-a-primitive-type-primitives-types-too

References - Books

- Java Fundamentals: Programming Basics for Beginners (2018)
- Fundamentals of Java Programming (2018)
- Java for Absolute Beginners: Learn to Program the Fundamentals the Java 9+ Way (2018)
- Java Programming for Beginners: Learn the fundamentals of programming with Java (2017)

Thanks!...