Xideral

Academia Java

Semana 1 -Dia 4

Herencia y Polimorfismo.

Presentado por:

Edgar Itzak Sánchez Rogers

Monterrey Nuevo León México a 15 de agosto de 2024

Introducción:

La herencia es un concepto en la programación orientada a objetos que consiste en compartir atributos y métodos a toda aquella clase que herede de una superclase. Una superclase es una clase que modela el común denominador de todas las clases que heredan de ella. Una subclase es una clase que obtiene los atributos de una superclase, las subclases pueden diferenciase entre ellas al tener atributos y métodos diferentes e independientes de otras subclases. Para indicar que una clase hereda de otra clase es necesario usar la palabra reservada “extends” y posteriormente el nombre de la superclase, ejemplo:

```
public class Teacher extends Employee
```

Las subclases pueden y suelen tener métodos en común, sin embargo, estos pueden variar en su comportamiento, esto ocasiona que distintos objetos a los que se les aplica la misma función pueden tener comportamientos diferentes, a esta característica se le conoce como Polimorfismo.

Demostración de herencia y polimorfismo en Java:

A continuación, se muestra un código en Java de demostración que implementa el concepto de herencia y polimorfismo:

Superclase (Product):

```java
package com.edgaritzak.product;

public abstract class product {
    private static int counter = 0; //to assign an auto incremental id
    protected int id;
    protected String name;
    protected Double price;

    //constructor
    public product(String name, Double price) {
        counter++;
        this.id = counter;
        this.name = name;
        this.price = price;
    }

    //methods
    void applyDiscount(int percentage) {
        price =price*(100-percentage)*.01;
    };

    //abstract methods
    abstract double getfinalPrice();
    abstract String verifyCaducity();

    //getters and setters
    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }
```

Subclase (CannedGoods)

```java
package com.edgaritzak.product;
import java.time.LocalDate;

public class CannedGoods extends product {
    private String brand;
    private LocalDate expirationDate;
    private String type;

    //constructor
    public CannedGoods(String name, Double price, String brand, LocalDate expirationDate, String type) {
        super(name, price);
        this.brand = brand;
        this.expirationDate = expirationDate;
        this.type = type;
    }

    //methods
    @Override
    double getfinalPrice() {
        // 5% tax
        return price*1.05;
    }

    @Override
    String verifyCaducity() {
        //Compare expiration date vs todays' date
        LocalDate todaysDate = LocalDate.now();
        if (expirationDate.isBefore(todaysDate)) {
            return "This product has not expired";
        } else {
            return "Warning: this product has already expired";
        }
    }
```

Subclase (Fruits_Vegetables)

```java
package com.edgaritzak.product;

public class Fruits_Vegetables extends product{
    private String type;
    private double kilograms;

    //constructor
    public Fruits_Vegetables(String name, Double price, String type, double kilograms) {
        super(name, price);
        this.type = type;
        this.kilograms = kilograms;
    }

    //methods
    @Override
    double getfinalPrice() {
         //tax free
        return kilograms*price;
    }

    @Override
    String verifyCaducity() {
        return "Please check product freshness";
    }

    @Override
    public String toString() {
        return "Fruits_Vegetables [id=" + id + ", name=" + name + ", price=" + price + ", type=" + type + ", kilograms="
                + kilograms + "]";
    }

```

Subclase (HouseholdAppliances)

```java
package com.edgaritzak.product;

public class HouseholdAppliances extends product{
    private String brand;
    private String model;
    private String type;

    //constructor
    public HouseholdAppliances(String name, Double price, String brand, String model, String type) {
        super(name, price);
        this.brand = brand;
        this.model = model;
        this.type = type;
    }

    //methods
    @Override
    double getfinalPrice(){
        // 20% tax
        return price *1.20;
    }

    @Override
    String verifyCaducity() {
        return "N/A";
    }

    @Override
    public String toString() {
        return "HouseholdAppliances [id=" + id + ", name=" + name + ", price=" + price + ", brand=" + brand + ",
                + model + ", type=" + type + "]";
    }
```

Como se puede observar, cada producto tiene diferentes atributos adicionales además de los atributos comunes (id, nombre, precio) además de diferentes comportamientos en el método getFinalPrice() ya que cada tipo de producto se le agrega un impuesto diferente, por último, cada producto tiene diferente comportamiento en el método verifyCaducity() para los electrodomésticos regresa “N/A”, para los productos frescos regresa la leyenda “Verificar frescura de producto” y para los productos enlatados hace una comparación entre la fecha de expiración y la fecha actual y regresa el resultado.

Clase(Main)

```java
package com.edgaritzak.product;

import java.time.LocalDate;
public class Main {

    public static void main(String[] args) {

        //create an array with products
        product[] productArray = {
            new Fruits_Vegetables("Ocean Spray Green Seedless Grapes", 2.99, "Grapes", 1.19),
            new HouseholdAppliances("Refrigerator French door 25ft", 1199.99, "Whirlpool", "WD5620S", "Refrigerator"),
            new CannedGoods("Heinz Baked Beans - 13.7oz – Jolly Posh Foods", 1.50, "Heinz", LocalDate.parse("2025-08-14"
            new CannedGoods("Del Monte Canned Fresh Cut Golden Sweet Whole Kernel Corn", 1.19, "Heinz", LocalDate.parse(
            new Fruits_Vegetables("Golden Delicious Apples | Stemilt", 1.99, "Apples", 2.53)
        };

        //get info of each product in the array
        for(product p:productArray) {
            //print info
            System.out.println(p.getName()+"\n - Caducity status: "+
            p.verifyCaducity()+"\n - Price before taxes: $"+ p.getPrice()+"\n - Total price (with tax): $"+
            p.getfinalPrice());

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

}
```

Ejecución:

```
Ocean Spray Green Seedless Grapes
 - Caducity status: Please check product freshness
 - Price before taxes: $2.99
 - Total price (with tax): $3.5581
------------------------
Refrigerator French door 25ft
 - Caducity status: N/A
 - Price before taxes: $1199.99
 - Total price (with tax): $1439.988
------------------------
Heinz Baked Beans - 13.7oz – Jolly Posh Foods
 - Caducity status: Warning: this product has already expired
 - Price before taxes: $1.5
 - Total price (with tax): $1.5750000000000002
------------------------
Del Monte Canned Fresh Cut Golden Sweet Whole Kernel Corn
 - Caducity status: This product has not expired
 - Price before taxes: $1.19
 - Total price (with tax): $1.2495
------------------------
Golden Delicious Apples | Stemilt
 - Caducity status: Please check product freshness
 - Price before taxes: $1.99
 - Total price (with tax): $5.0347
------------------------
```

Conclusión:

El uso de herencia y polimorfismo ayuda a crear un código más versátil, escalable y flexible por lo que mejora la posibilidad de reutilizar software ahorrando tiempo y esfuerzo en modificar implementaciones nuevas. La herencia nos permite modelar un programa pensando en conceptos abstractos y agrupando objetos y comportamientos, esto permite crear códigos más compactos y versátiles.

Referencias:

[1] Herencia (informática) - Wikipedia, la enciclopedia libre

[2 Introducción a POO en Java: Herencia y polimorfismo | OpenWebinars

[3] Polimorfismo en Java: Programación orientada a objetos | ifGeekThenNTTDATA